

Douglas R. Stinson
Stafford Tavares (Eds.)

LNCS 2012

Selected Areas in Cryptography

**7th Annual International Workshop, SAC 2000
Waterloo, Ontario, Canada, August 2000
Proceedings**



Springer

Lecture Notes in Computer Science
Edited by G. Goos, J. Hartmanis and J. van Leeuwen

2012

Preface

SAC 2000 was the seventh in a series of annual workshops on Selected Areas in Cryptography. Previous workshops were held at Queen's University in Kingston (1994, 1996, 1998, and 1999) and at Carleton University in Ottawa (1995 and 1997). The intent of the workshops is to provide a relaxed atmosphere in which researchers in cryptography can present and discuss new work on selected areas of current interest.

The themes for the SAC 2000 workshop were:

- design and analysis of symmetric key cryptosystems,
- primitives for private key cryptography, including block and stream ciphers, hash functions, and MACs,
- efficient implementations of cryptographic systems in public and private key cryptography,
- cryptographic solutions for web/internet security.

A total of 41 papers were submitted to SAC 2000, one of which was subsequently withdrawn. After a review process that had all papers reviewed by at least 3 referees, 24 papers were accepted for presentation at the workshop. As well, we were fortunate to have the following two invited speakers at SAC 2000:

- M. Bellare, UCSD (U.S.A.)
“The Provable-Security Approach to Authenticated Session-Key Exchange”
- D. Boneh, Stanford U. (U.S.A.)
“Message Authentication in a Multicast Environment”

The program committee for SAC 2000 consisted of the following members: L. Chen, H. Heys, L. Knudsen, S. Moriai, L. O'Connor, D. Stinson, S. Tavares, S. Vaudenay, A. Youssef, and R. Zuccherato. Many thanks are due to the program committee for their hard work. Also, Amr Youssef provided great assistance in making the reviewing process run smoothly.

We are appreciative of the financial support provided by Certicom Corporation, CITO, Entrust Technologies, MITACS, and the University of Waterloo. Special thanks are due to Frances Hannigan, who was responsible for the local arrangements, and for making sure that everything ran smoothly during the workshop. Fran also assisted in preparing the workshop proceedings. Many people helped in the reviewing process by acting as sub-referees, and we appreciate all their help. Finally, we thank all the workshop participants for making SAC 2000 a success.

March 2001

Doug Stinson
Stafford Tavares

Springer

Berlin

Heidelberg

New York

Barcelona

Hong Kong

London

Milan

Paris

Singapore

Tokyo

Douglas R. Stinson Stafford Tavares (Eds.)

Selected Areas in Cryptography

7th Annual International Workshop, SAC 2000
Waterloo, Ontario, Canada, August 14-15, 2000
Proceedings



Springer

Series Editors

Gerhard Goos, Karlsruhe University, Germany
Juris Hartmanis, Cornell University, NY, USA
Jan van Leeuwen, Utrecht University, The Netherlands

Volume Editors

Douglas R. Stinson
University of Waterloo
Department of Combinatorics and Optimization
Waterloo, Ontario, N2L 3G1, Canada
E-mail: dstinson@uwaterloo.ca

Stafford Tavares
Queen's University
Department of Electrical and Computer Engineering
Kingston, Ontario, K7L 3N6
E-mail: tavares@cc.queensu.ca

Cataloging-in-Publication Data applied for

Die Deutsche Bibliothek - CIP-Einheitsaufnahme

Selected areas in cryptography : 7th annual international workshop ;
proceedings / SAC 2000, Waterloo, Ontario, Canada, August 14 - 15,
2000. Douglas R. Stinson ; Stafford Tavares (ed.). - Berlin ;
Heidelberg ; New York ; Barcelona ; Hong Kong ; London ; Milan ;
Paris ; Singapore ; Tokyo : Springer, 2001
(Lecture notes in computer science ; Vol. 2012)
ISBN 3-540-42069-X

CR Subject Classification (1998): E.3, C.2, D.4.6, K.6.5, F.2.1-2, H.4.3

ISSN 0302-9743

ISBN 3-540-42069-X Springer-Verlag Berlin Heidelberg New York

This work is subject to copyright. All rights are reserved, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, re-use of illustrations, recitation, broadcasting, reproduction on microfilms or in any other way, and storage in data banks. Duplication of this publication or parts thereof is permitted only under the provisions of the German Copyright Law of September 9, 1965, in its current version, and permission for use must always be obtained from Springer-Verlag. Violations are liable for prosecution under the German Copyright Law.

Springer-Verlag Berlin Heidelberg New York
a member of BertelsmannSpringer Science+Business Media GmbH

<http://www.springer.de>

© Springer-Verlag Berlin Heidelberg 2001
Printed in Germany

Typesetting: Camera-ready by author, data conversion by Olgun Computergrafik Heidelberg
Printed on acid-free paper SPIN 10782303 06/3142 5 4 3 2 1 0

Organization

Program Committee

| | |
|-----------------------|-------------------------------------|
| D. Stinson (co-chair) | University of Waterloo |
| S. Tavares (co-chair) | Queen's University at Kingston |
| L. Chen | Motorola (USA) |
| H. Heys | Memorial University of Newfoundland |
| L. Knudsen | University of Bergen |
| S. Moriai | NTT Labs. (Japan) |
| L. O'Connor | European Security COE (Switzerland) |
| S. Vaudenay | EPFL (Switzerland) |
| A. Youssef | University of Waterloo |
| R. Zuccherato | Entrust Technologies, Ottawa |

Local Organizing Committee

| | |
|------------------|--------------------------------|
| Doug Stinson | University of Waterloo |
| Stafford Tavares | Queen's University at Kingston |
| Frances Hannigan | University of Waterloo |

Sponsoring Institutions

Certicom Corporation
CITO
Entrust Technologies
MITACS
University of Waterloo

Table of Contents

Cryptanalysis I

| | |
|--|----|
| Analysis of IS-95 CDMA Voice Privacy | 1 |
| <i>Muxiang Zhang, Christopher Carroll, and Agnes Chan</i> | |
| Attacks on Additive Encryption of Redundant Plaintext and Implications on Internet Security | 14 |
| <i>David A. McGrew and Scott R. Fluhrer</i> | |
| Cryptanalysis of the “Augmented Family of Cryptographic Parity Circuits” Proposed at ISW’97 | 29 |
| <i>A.M. Youssef</i> | |

Block Ciphers – New Designs

| | |
|---|----|
| <i>Camellia</i> : A 128-Bit Block Cipher Suitable for Multiple Platforms – Design and Analysis | 39 |
| <i>Kazumaro Aoki, Tetsuya Ichikawa, Masayuki Kanda, Mitsuru Matsui, Shiho Moriai, Junko Nakajima, and Toshio Tokita</i> | |
| DFCv2 | 57 |
| <i>Louis Granboulan, Phong Q. Nguyen, Fabrice Noilhan, and Serge Vaudenay</i> | |
| The Block Cipher Hierocrypt | 72 |
| <i>Kenji Ohkuma, Hirofumi Muratani, Fumihiko Sano, and Shinichi Kawamura</i> | |
| Symmetric Block Ciphers Based on Group Bases | 89 |
| <i>Valér Čanda, Tran van Trung, Spyros Magliveras, and Tamás Horváth</i> | |

Elliptic Curves and Efficient Implementations

| | |
|--|-----|
| Speeding up the Arithmetic on Koblitz Curves of Genus Two | 106 |
| <i>Christian Günther, Tanja Lange, and Andreas Stein</i> | |
| On Complexity of Polynomial Basis Squaring in \mathbb{F}_{2^m} | 118 |
| <i>Huapeng Wu</i> | |

Security Protocols and Applications

| | |
|---|-----|
| Dynamic Multi-threshold Metering Schemes | 130 |
| <i>Carlo Blundo, Annalisa De Bonis, Barbara Masucci, and Douglas R. Stinson</i> | |

VIII Table of Contents

| | |
|--|-----|
| Chained Stream Authentication | 144 |
| <i>Francesco Bergadano, Davide Cavagnino, and Bruno Crispo</i> | |

| | |
|--|-----|
| A Global PMI for Electronic Content Distribution | 158 |
| <i>Carlisle Adams and Robert Zuccherato</i> | |

Block Ciphers and Hash Functions

| | |
|---|-----|
| A Polynomial-Time Universal Security Amplifier in the Class of Block Ciphers | 169 |
| <i>John O. Pliam</i> | |

| | |
|---|-----|
| Decorrelation over Infinite Domains: The Encrypted CBC-MAC Case | 189 |
| <i>Serge Vaudenay</i> | |

| | |
|--|-----|
| HAS-V: A New Hash Function with Variable Output Length | 202 |
| <i>Nan Kyoung Park, Joon Ho Hwang, and Pil Joong Lee</i> | |

Boolean Functions and Stream Ciphers

| | |
|--|-----|
| On Welch-Gong Transformation Sequence Generators | 217 |
| <i>G. Gong and A.M. Youssef</i> | |

| | |
|--|-----|
| Modes of Operation of Stream Ciphers | 233 |
| <i>Jovan Dj. Golić</i> | |

| | |
|---|-----|
| LILI Keystream Generator | 248 |
| <i>Leonie Ruth Simpson, E. Dawson, Jovan Dj. Golić, and William L. Millan</i> | |

| | |
|--|-----|
| Improved Upper Bound on the Nonlinearity of High Order Correlation Immune Functions | 262 |
| <i>Yuliang Zheng and Xian-Mo Zhang</i> | |

Public Key Systems

| | |
|--|-----|
| Towards Practical Non-interactive Public Key Cryptosystems Using Non-maximal Imaginary Quadratic Orders | 275 |
| <i>Detlef Hühnlein, Michael J. Jacobson, Jr., and Damian Weber</i> | |

| | |
|---|-----|
| On the Implementation of Cryptosystems Based on Real Quadratic Number Fields | 288 |
| <i>Detlef Hühnlein and Sachar Paulus</i> | |

Cryptanalysis II

| | |
|--|-----|
| Root Finding Interpolation Attack | 303 |
| <i>Kaoru Kurosawa, Tetsu Iwata, and Viet Duong Quang</i> | |

| | |
|--|-----|
| Differential Cryptanalysis of Reduced Rounds of GOST | 315 |
| <i>Haruki Seki and Toshinobu Kaneko</i> | |
| Practical Security Evaluation against Differential and Linear Cryptanalyses for Feistel Ciphers with SPN Round Function . . | 324 |
| <i>Masayuki Kanda</i> | |
| Author Index | 339 |

Analysis of IS-95 CDMA Voice Privacy

Muxiang Zhang¹, Christopher Carroll¹, and Agnes Chan²

¹ GTE Laboratories Inc., 40 Sylvan Road LA0MS59, Waltham, MA 02451
{mzhang,ccarroll}@gte.com

² College of Computer Science, Northeastern University, Boston, MA 02115
ahchan@ccs.neu.edu

Abstract. The voice privacy of IS-95 CDMA cellular system is analyzed in this paper. By exploiting information redundancy on the downlink traffic channel, it is shown that an eavesdropper can recover the voice privacy mask after eavesdropping the transmission on the downlink traffic channel for about one second. Thus, IS-95 CDMA voice privacy is vulnerable under ciphertext-only attacks.

1 Introduction

IS-95 *Code Division Multiple Access* (CDMA) is an interim industry standard [3] for cellular communication systems. In IS-95 CDMA, a pseudo-random pattern, which is a high bit-rate binary sequence known as *long code sequence*, is added to the low bit-rate voice signal. Adding a high bit-rate noise-like signal to a voice signal makes the voice signal more robust and less susceptible to interference. It enables low-power transmission to take place, resulting in cheaper mobile stations with long-lasting battery life. The long code sequence, which is only known to the designated receiver, is also expected to provide certain level of privacy to the voice signal. To decode the voice signal, the eavesdropper has to recover the long code sequence from the intercepted signal. The long code sequence is generated by the *long code generator* as shown in Figure 3. The long code generator consists of a 42-bit number called *long code mask* and a 42-bit linear feedback shift register (LFSR) specified by the following characteristic polynomial:

$$x^{42} + x^{35} + x^{33} + x^{31} + x^{27} + x^{26} + x^{25} + x^{22} + x^{21} + x^{19} \\ + x^{18} + x^{17} + x^{16} + x^{10} + x^7 + x^6 + x^5 + x^3 + x^2 + x + 1.$$

The inner product of the LFSR state and the long code mask produces the long code sequence.

Voice privacy of IS-95 CDMA is provided by means of the long code mask. The long code mask is not transmitted through any channel, it is constructed by the base station and the mobile station. To recover the long code sequence, the eavesdropper may exhaustively search the 42-bit long code mask, with a time complexity of $O(2^{42})$. This attack is viable but is hard to implement in real time. Alternatively, it can be shown that the long code sequence can also

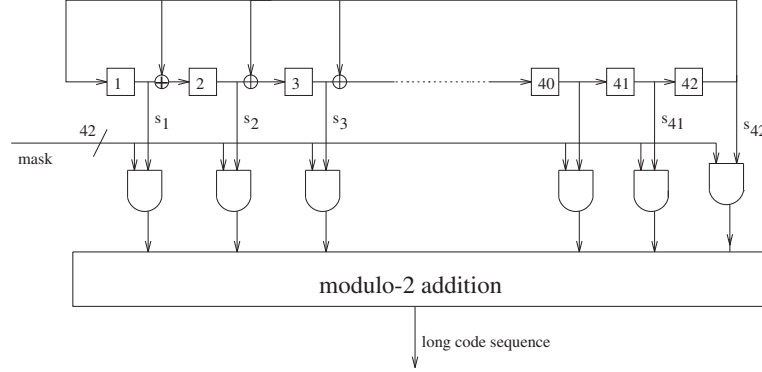


Fig. 1. The long code generator

be recovered if the eavesdropper can obtain 42 bits of plaintext-ciphertext pairs. As there are many mobile stations transmitting simultaneously on the traffic channel and each mobile station only transmits approximately 3 minutes on the average, it is rather difficult to obtain 42 bits of the plaintext message. In this paper, we investigate the security of IS-95 CDMA voice privacy under ciphertext-only attacks. By exploiting information redundancy introduced by channel coding, it is shown that the eavesdropper can recover the voice privacy mask after eavesdropping the transmission on the traffic channel for about one second. This paper is organized as follows. Section 2 gives an equivalent long code generator which can simplify the analysis of CDMA voice privacy. Section 3 describes two kinds of ciphertext-only attacks and Section 4 presents some concluding remarks.

2 An Equivalent Long Code Generator

Let $(m_1, m_2, \dots, m_{42})$ denote the 42-bit mask and $(s_1(k), s_2(k), \dots, s_{42}(k))$ denote the state of the LFSR at time instant k . Then the long code sequence $c(k)$ at time instant k can be represented as

$$c(k) = m_1 s_1(k) + m_2 s_2(k) + \dots + m_{42} s_{42}(k), \quad (1)$$

where the addition is the modulo-2 addition. Since $s_1(k), s_2(k), \dots, s_{42}(k)$ are the outputs of the 42 stages of the LFSR, they are the same sequence but only differ in the phase. So, for any i , $1 \leq i \leq 42$, $s_i(k)$ satisfies the following linear recurrence equation,

$$s_i(k) = a_1 s_i(k-1) + a_2 s_i(k-2) + \dots + a_{42} s_i(k-42), \quad (2)$$

where a_i is the coefficient of x^i in the characteristic polynomial of the LFSR. Substituting (2) into (1), we have

$$\begin{aligned}
c(k) &= \sum_{i=1}^{42} m_i s_i(k) \\
&= \sum_{i=1}^{42} m_i \left(\sum_{j=1}^{42} a_j s_i(k-j) \right) \\
&= \sum_{i=1}^{42} a_i \left(\sum_{j=1}^{42} m_j s_j(k-i) \right).
\end{aligned}$$

By equation (1),

$$\sum_{j=1}^{42} m_j s_j(k-i) = c(k-i).$$

Hence, it follows that

$$c(k) = a_1 c(k-1) + a_2 c(k-2) + \cdots + a_{42} c(k-42), \quad (3)$$

which means that the long code sequence is also generated by the same LFSR. Although every mobile station uses a different long code mask, their long code sequences are the same except for the different phases. The long code mask only affects the phase of the long code sequence.

Let $C(k) = (c(k), c(k-1), \dots, c(k-41))$ and $C(k+1) = (c(k+1), c(k), \dots, c(k-42))$ be two consecutive states of the LFSR. By equation (3), we have

$$\begin{bmatrix} c(k+1) \\ c(k) \\ \vdots \\ \vdots \\ c(k-42) \end{bmatrix} = \begin{bmatrix} a_1 & a_2 & a_3 & \cdots & a_{40} & a_{41} & a_{42} \\ 1 & 0 & 0 & & 0 & 0 & 0 \\ 0 & 1 & 0 & & 0 & 0 & 0 \\ 0 & 0 & 1 & & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 1 & 0 & 0 \\ 0 & 0 & 0 & \cdots & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} c(k) \\ c(k-1) \\ \vdots \\ \vdots \\ c(k-41) \end{bmatrix}. \quad (4)$$

Let A^t denote the 42×42 matrix of the above equation and a^t denote the vector $(a_1, a_2, \dots, a_{42})$, where A^t is the transpose of A . Then for any $n > k$,

$$C(n) = C(k) A^{n-k}.$$

By equation (4), it follows that

$$c(n) = C(k) A^{n-k-1} a. \quad (5)$$

Equation (5) indicates that the state $C(k)$ of the LFSR can be calculated if any 42 bits of the long code sequence are known. Since the LFSR generating the long code sequence is publicly known, the coefficients $a_i, 1 \leq i \leq 42$ are available to the eavesdropper. Thus, the long code sequence can be generated if the eavesdropper can recover 42 bits of the long code sequence.

3 Attacks on CDMA Voice Privacy

Before the analysis of IS-95 CDMA voice privacy, let us give a simple description of CDMA channels. The CDMA channels consists of traffic channels and control channels. The traffic channels carry user information while the control channels carry signalling information. The traffic channels can be further divided into downlink and uplink traffic channels.

3.1 CDMA Traffic Channels

The downlink channel carries information from the base station to the mobile station. On the downlink traffic channel, the vocoder accepts the voice signal and produces a compressed data stream. IS-95 CDMA specifies a variable-rate vocoder operating at full, $\frac{1}{2}$, $\frac{1}{4}$, and $\frac{1}{8}$ rates. The rate is determined according to the power level of the background noise. There are currently two types of vocoders: the one operating in a 9.6-kbps, and the other operating in a 14.4-kbps, referred to as rate set 1 and rate set 2, respectively. Rate set 1 contains four elements: 9.6, 4.8, 2.4, and 1.2 kbps. Rate set 2 also contains four elements: 14.4, 7.2, 3.6, and 1.8 kbps. The mobile station has to support rate set 1, while rate set 2 is optional.

The data stream from the vocoder is structured in 20-ms frames. The full rate of rate set 1 vocoder is 8.6 kbps, which generates 172 bits every 20 ms. The frame quality indicator, which is cyclic redundancy checking (CRC) digits derived from the 172 information bits, is added to the 172 bits along with an 8-bit tail (set to 0). The 9.6 kbps frame is a result of 192 bits ($172+12+8$) transmitted every 20 ms. The 4.8-kbps frame has the same structure, while 2.4- and 1.2-kbps frames do not have frame quality indicator fields since most of the information sent in these frames is background noise. Rate set 2 frames have similar structures as rate set 1. The 20-ms data frames are encoded, interleaved, scrambled, spread, and modulated before they are sent onto the air interface. Figure 2 shows the different functions which act on rate set 1 data frames.

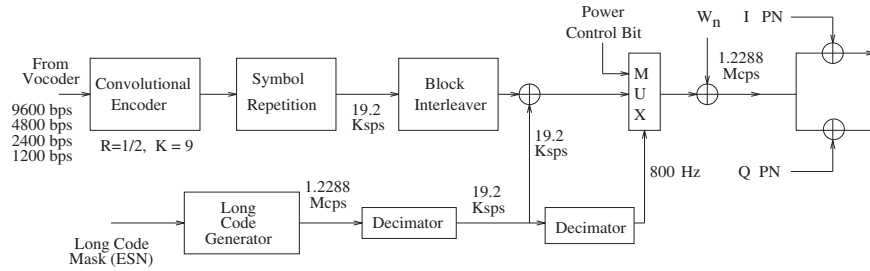


Fig. 2. Rate set 1 downlink traffic channel generation

The convolutional encoder provides error-correction capability to the downlink traffic channels. The convolutional encoder replace each single input bit with

two bits (called code symbols). The symbol repeater repeats the code symbols produced by the convolutional encoder as necessary to result in an output stream with the fixed rate 19.2 ksps (kilo-symbols per second). For example, to achieve this, it does not repeat anything if the input rate is 19.2 ksps. It repeats each symbol twice if the input rate is 9.6 ksps, it repeats each symbol 4 times if the input rate is 4.8 ksps. The block Interleaver shuffles the code symbols in each data frame. Data scrambling is provided through the long code generator. A power control subchannel is continuously transmitted on the downlink traffic channel. It is used to control the mobile station's power on the uplink. This subchannel transmits at a rate of one bit every 1.25 ms (i.e., 800 bps). The power control bit which is two symbols long replaces two consecutive symbols on the downlink traffic channel. The 19.2 ksps data stream, which has been punctured with the power control bits of the power control subchannel, is spread by a Walsh code. Following the Walsh code spreading, the data is modulated for transmission.

Like the downlink traffic channel, the uplink traffic channel also supports two rate sets, depending on the type of vocoder used. Figure 3 shows the overall structure of the uplink traffic channel for rate set 1.

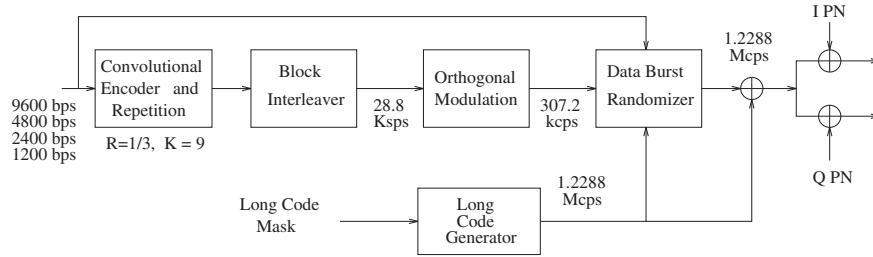


Fig. 3. Rate set 1 uplink traffic channel generation

3.2 Information Redundancy on the Downlink Traffic Channel

On the downlink traffic channel, information bits from the vocoder are coded by a convolution encoder as shown in Figure 4. The convolutional encoder has 1-bit input, 2-bit output, and 8-bit memory. Initially the 8 memory bits are filled with all 0s. For every bit of input, the convolutional encoder outputs 2 bits. In coding theory, each bit of output is called a code symbol. Such a convolutional encoder is called a half-rate convolutional encoder [1]. Because of the 8-bit memory, each code symbol is related to 9 information bits. The number 9 is called the constraint length of the convolutional encoder. The half-rate convolutional encoder with constraint length of 9 is also called a (2, 1, 8) convolutional encoder.

Let $b = (b_0, b_1, b_2, \dots)$ denote the input sequence entering the convolutional encoder. The two output sequences

$$v^{(1)} = (v_0^{(1)}, v_1^{(1)}, v_2^{(1)}, \dots)$$

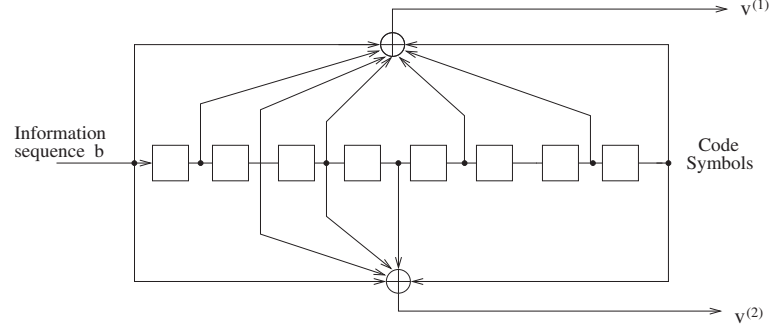


Fig. 4. Half-rate convolutional encoder

and

$$v^{(2)} = (v_0^{(2)}, v_1^{(2)}, v_2^{(2)}, \dots)$$

can be computed as follows:

$$v_l^{(1)} = b_l + b_{l-1} + b_{l-2} + b_{l-3} + b_{l-5} + b_{l-7} + b_{l-8}, \quad (6)$$

$$v_l^{(2)} = b_l + b_{l-2} + b_{l-3} + b_{l-4} + b_{l-8}. \quad (7)$$

where $b_{l-i} = 0$ for all $l < i$. The two output sequences are multiplexed into a single sequence, called the code word. Let $v = (v_0^{(1)}, v_0^{(2)}, v_1^{(1)}, v_1^{(2)}, v_2^{(1)}, v_2^{(2)}, \dots)$ denote the code word. From (6) and (7), the code word satisfies the following equation:

$$vH = 0, \quad (8)$$

where H is a semi-infinite matrix given by

$$H = \begin{bmatrix} 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 0 & 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 0 & 1 & 0 & 1 & 1 \\ & & & & \vdots & \vdots & & & \\ & & & & & 1 & 0 & \dots & \\ & & & & & 1 & 1 & \dots & \\ & & & & & & & \ddots & \end{bmatrix}.$$

In equation (8), the code word is treated as a semi-infinite sequence since the input sequence to the convolutional encoder may be semi-infinite. When the input sequence to the convolutional encoder is truncated to k bits and the last 8 bits are all zero, the code symbol sequence has length $n = 2k$ and satisfies the following equation:

$$v[H]_n = 0, \quad (9)$$

where $[H]_n$ is composed of the first n rows and $n/2 + 8$ columns of H .

Recall that the information bits entering the convolutional encoder are structured in 20-ms frames and the last 8 bits of each frame are set to zero. As a consequence, the code symbols in every 20-ms frame satisfy equation (9). Thus, every frame on the downlink traffic channel contains redundant information. The redundant information can be used to solve for the long code sequence.

Except the convolutional encoding, symbol repetition also incurs redundant information. The code symbols from the convolutional encoder are repeated whenever the information rate is lower than 9.6 kbps. Every code symbol at the 4.8 kbps rate is repeated 1 time, every code symbol at the 2.4 kbps rate is repeated 3 times, and every code symbol at the 1.2 kbps rate is repeated 7 times. Every frame contains 384 code symbols after repetition. Let $u = (u_1, u_2, \dots, u_{384})$ denote the frame after code symbol repetition. If each code symbol appears two times in u , it is obvious that

$$uE_2 = 0, \quad (10)$$

where E_2 is a matrix described by

$$E_2 = \begin{bmatrix} 1 & 0 & 0 & \cdots & 0 \\ 1 & 0 & 0 & \cdots & 0 \\ 0 & 1 & 0 & & 0 \\ 0 & 1 & 0 & & 0 \\ & & & \ddots & \\ 0 & 0 & 0 & \cdots & 1 \\ 0 & 0 & 0 & \cdots & 1 \end{bmatrix}.$$

If we sample every other code symbol in $u = (u_1, u_2, \dots, u_{384})$, the sampled code symbols $(u_1, u_3, \dots, u_{383})$ should satisfy equation (9) since $(u_1, u_3, \dots, u_{383})$ are the output symbols of the convolutional encoder. Mathematically, the sampling of every other symbols of $u = (u_1, u_2, \dots, u_{384})$ can be described by the matrix multiplication uD_2 , where

$$D_2 = \begin{bmatrix} 1 & 0 & 0 & \cdots & 0 \\ 0 & 0 & 0 & & 0 \\ 0 & 1 & 0 & & 0 \\ \vdots & & & \ddots & \\ 0 & 0 & 0 & & 1 \\ 0 & 0 & 0 & \cdots & 0 \end{bmatrix}.$$

Substituting v in (9) by uD_2 , we have

$$uD_2[H]_{192} = 0. \quad (11)$$

Generally, if each code symbol in $u = (u_1, u_2, \dots, u_{384})$ appears k times, where $k = 1, 2, 4, 8$, then the code symbol frame $u = (u_1, u_2, \dots, u_{384})$ satisfies the following equations:

$$uE_k = 0, \quad (12)$$

$$uD_k[H]_{384/k} = 0, \quad (13)$$

where $E_1 = 0$, $D_1 = \mathbf{I}$, E_4 , D_4 , E_8 , and D_8 can be derived in similar ways as E_2 and D_2 .

3.3 Recovery of the Long Code Sequence

To recover the long code sequence, the eavesdropper intercepts the downlink traffic channel, demodulates the intercepted data frames, and despreads the data frames using the Walsh code specific to the channel. Let $r = (r_1, r_2, \dots, r_{384})$ denote the data frame after despreading. Correspondingly, let $c = (c(t_1), c(t_2), \dots, c(t_{384}))$ denote the 384 bits of the long code sequence used for scrambling, $s = (s_1, s_2, \dots, s_{384})$ denote the output of the block interleaver, and $u = (u_1, u_2, \dots, u_{384})$ denote the input to the block interleaver. Figure 5 describes the relationship among these notations on the downlink traffic channel.

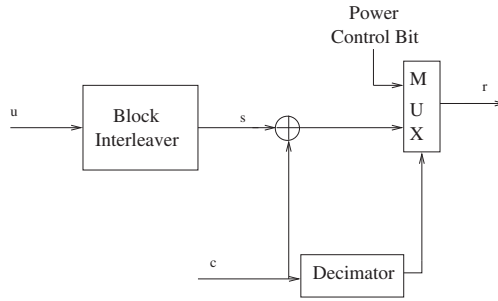


Fig. 5. Block interleaving and scrambling

The block interleaver is a permutation of the 384 input code symbols. The input symbols are entered as a 24×16 array and the interleaver produces a 24×16 output array. Table 1 describes the input array. The table is read down by columns from the left to the right. That is, the first input symbol u_1 is at the top left, the second input symbol u_2 is just below the first input symbol, and the 25th input symbol u_{25} is just to the right of the first input symbol. The output array is given by Table 2, which is read the same way as Table 1, that is, the first output symbol is u_1 , the second output symbol is u_{65} , and the 25th output symbol is u_9 .

Mathematically, the block interleaver can be represented by a permutation matrix P , namely,

$$s = uP, \quad (14)$$

where P is a 384×384 matrix, each row and column has only one 1.

For the moment, let's assume that the power control bits were not transmitted through the downlink traffic channel. Then we have

$$r = s \oplus c. \quad (15)$$

Table 1. Downlink traffic channel interleaver input

| | | | | | | | | | | | | | | | |
|----|----|----|----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|
| 1 | 25 | 49 | 73 | 97 | 121 | 145 | 169 | 193 | 217 | 241 | 165 | 289 | 313 | 337 | 361 |
| 2 | 26 | 50 | 74 | 98 | 122 | 246 | 170 | 194 | 218 | 242 | 266 | 290 | 314 | 338 | 362 |
| 3 | 27 | 51 | 75 | 99 | 123 | 147 | 171 | 195 | 219 | 243 | 267 | 291 | 315 | 339 | 363 |
| 4 | 28 | 52 | 76 | 100 | 124 | 148 | 172 | 196 | 220 | 244 | 268 | 292 | 316 | 340 | 364 |
| 5 | 29 | 53 | 77 | 101 | 125 | 149 | 173 | 197 | 221 | 245 | 269 | 293 | 317 | 341 | 365 |
| 6 | 30 | 54 | 78 | 102 | 126 | 150 | 174 | 198 | 222 | 246 | 270 | 294 | 318 | 342 | 366 |
| 7 | 31 | 55 | 79 | 103 | 127 | 151 | 175 | 199 | 223 | 247 | 271 | 295 | 319 | 343 | 367 |
| 8 | 32 | 56 | 80 | 104 | 128 | 152 | 176 | 200 | 224 | 248 | 272 | 296 | 320 | 344 | 368 |
| 9 | 33 | 57 | 81 | 105 | 129 | 153 | 177 | 201 | 225 | 249 | 273 | 297 | 321 | 345 | 369 |
| 10 | 34 | 58 | 81 | 106 | 130 | 154 | 178 | 202 | 226 | 250 | 274 | 298 | 322 | 346 | 370 |
| 11 | 35 | 59 | 83 | 107 | 131 | 155 | 179 | 203 | 227 | 251 | 275 | 299 | 323 | 347 | 371 |
| 12 | 36 | 60 | 84 | 108 | 132 | 156 | 180 | 204 | 228 | 252 | 276 | 300 | 324 | 348 | 372 |
| 13 | 37 | 61 | 85 | 109 | 133 | 157 | 181 | 205 | 229 | 253 | 277 | 301 | 325 | 349 | 373 |
| 14 | 38 | 62 | 86 | 110 | 134 | 158 | 182 | 206 | 230 | 254 | 278 | 302 | 326 | 350 | 374 |
| 15 | 39 | 63 | 87 | 111 | 135 | 159 | 183 | 207 | 231 | 255 | 279 | 303 | 327 | 351 | 375 |
| 16 | 40 | 64 | 88 | 112 | 136 | 160 | 184 | 208 | 232 | 256 | 280 | 304 | 328 | 352 | 376 |
| 17 | 41 | 65 | 89 | 113 | 137 | 161 | 185 | 209 | 233 | 257 | 281 | 305 | 329 | 353 | 377 |
| 18 | 42 | 66 | 90 | 114 | 138 | 162 | 186 | 210 | 234 | 258 | 282 | 306 | 330 | 354 | 378 |
| 19 | 43 | 67 | 91 | 115 | 139 | 163 | 187 | 211 | 235 | 259 | 283 | 307 | 331 | 355 | 379 |
| 20 | 44 | 68 | 92 | 116 | 140 | 164 | 188 | 212 | 236 | 260 | 284 | 308 | 332 | 356 | 380 |
| 21 | 45 | 69 | 93 | 117 | 141 | 165 | 189 | 213 | 237 | 261 | 285 | 309 | 333 | 357 | 381 |
| 22 | 46 | 70 | 94 | 118 | 142 | 166 | 190 | 214 | 238 | 262 | 286 | 310 | 334 | 358 | 382 |
| 23 | 47 | 71 | 95 | 119 | 143 | 167 | 191 | 215 | 239 | 263 | 287 | 311 | 335 | 359 | 383 |
| 24 | 48 | 72 | 96 | 120 | 144 | 168 | 192 | 216 | 240 | 264 | 288 | 312 | 336 | 360 | 384 |

Table 2. Downlink traffic channel interleaver output

| | | | | | | | | | | | | | | | |
|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|
| 1 | 9 | 5 | 13 | 3 | 11 | 7 | 15 | 2 | 10 | 6 | 14 | 4 | 12 | 8 | 16 |
| 65 | 73 | 69 | 77 | 67 | 75 | 71 | 79 | 66 | 74 | 70 | 78 | 68 | 76 | 72 | 80 |
| 129 | 137 | 133 | 141 | 131 | 139 | 135 | 143 | 130 | 138 | 134 | 142 | 132 | 140 | 136 | 144 |
| 193 | 201 | 197 | 205 | 195 | 203 | 199 | 207 | 194 | 202 | 198 | 206 | 196 | 204 | 200 | 208 |
| 257 | 265 | 261 | 269 | 259 | 267 | 263 | 271 | 258 | 266 | 262 | 270 | 260 | 268 | 264 | 272 |
| 321 | 329 | 325 | 333 | 323 | 331 | 327 | 335 | 322 | 330 | 326 | 334 | 324 | 332 | 328 | 336 |
| 33 | 41 | 37 | 45 | 35 | 43 | 39 | 47 | 34 | 42 | 38 | 46 | 36 | 44 | 40 | 48 |
| 97 | 105 | 101 | 109 | 99 | 107 | 103 | 111 | 98 | 106 | 102 | 110 | 100 | 108 | 104 | 112 |
| 161 | 169 | 165 | 173 | 163 | 171 | 167 | 175 | 162 | 170 | 166 | 174 | 164 | 172 | 168 | 176 |
| 225 | 233 | 229 | 237 | 227 | 235 | 231 | 239 | 226 | 234 | 230 | 238 | 228 | 236 | 232 | 240 |
| 289 | 297 | 293 | 301 | 291 | 299 | 295 | 303 | 290 | 298 | 294 | 302 | 292 | 300 | 296 | 304 |
| 353 | 361 | 357 | 365 | 355 | 363 | 359 | 367 | 354 | 362 | 358 | 366 | 356 | 364 | 360 | 368 |
| 17 | 25 | 21 | 29 | 19 | 27 | 23 | 31 | 18 | 26 | 22 | 30 | 20 | 28 | 24 | 32 |
| 81 | 89 | 85 | 93 | 83 | 91 | 87 | 95 | 82 | 90 | 86 | 94 | 84 | 92 | 88 | 96 |
| 145 | 153 | 149 | 157 | 147 | 155 | 151 | 159 | 146 | 154 | 150 | 158 | 148 | 156 | 152 | 160 |
| 209 | 217 | 213 | 221 | 211 | 219 | 215 | 223 | 210 | 218 | 214 | 222 | 212 | 220 | 216 | 224 |
| 273 | 281 | 277 | 285 | 275 | 283 | 279 | 287 | 274 | 282 | 278 | 286 | 276 | 284 | 280 | 288 |
| 337 | 345 | 341 | 349 | 339 | 347 | 343 | 351 | 338 | 346 | 342 | 350 | 340 | 348 | 344 | 352 |
| 49 | 57 | 53 | 61 | 51 | 59 | 55 | 63 | 50 | 58 | 54 | 62 | 52 | 60 | 56 | 64 |
| 113 | 121 | 117 | 125 | 115 | 123 | 119 | 127 | 114 | 122 | 118 | 126 | 116 | 124 | 120 | 128 |
| 177 | 185 | 181 | 189 | 179 | 187 | 183 | 191 | 178 | 186 | 182 | 190 | 180 | 188 | 184 | 192 |
| 241 | 249 | 245 | 253 | 243 | 251 | 247 | 255 | 242 | 250 | 246 | 254 | 244 | 252 | 248 | 256 |
| 305 | 313 | 309 | 317 | 307 | 315 | 311 | 319 | 306 | 314 | 310 | 318 | 308 | 316 | 312 | 320 |
| 369 | 377 | 373 | 381 | 371 | 379 | 375 | 383 | 370 | 378 | 374 | 382 | 372 | 380 | 376 | 384 |

From (14) and (15),

$$u = (r \oplus c)P^{-1}. \quad (16)$$

Substituting (16) into (12) and (13),

$$(r \oplus c)P^{-1}E_k = 0, \quad (17)$$

$$(r \oplus c)P^{-1}D_k[H]_{384/k} = 0, \quad (18)$$

where k is equal to the number of times each code symbol appears in the frame.

Let $C(t_0) = (c(t_0), c(t_0 - 1), \dots, c(t_0 - 41))$ denote the state of the LFSR that the eavesdropper would like to solve for. By (5), $c = (c(t_1), c(t_2), \dots, c(t_{384}))$ can be represented as

$$c = C(t_0)\hat{A}. \quad (19)$$

where $\hat{A} = (A^{t_1-t_0-1}, A^{t_2-t_0-1}, \dots, A^{t_{384}-t_0-1})a$.

Substituting (19) into (17) and (18), we have the following equations with $c(t_0), c(t_0 - 1), \dots, c(t_0 - 41)$ as variables,

$$(r \oplus C(t_0)\hat{A})P^{-1}E_k = 0, \quad (20)$$

$$(r \oplus C(t_0)\hat{A})P^{-1}D_k[H]_{384/k} = 0. \quad (21)$$

Corresponding to every k , $k = 1, 2, 4, 8$, there are 200 or more linear equations involved in (20) and (21). Solving these linear equations, the state of the LFSR can be recovered.

Now, taking into account of the power control bits, many linear equations in (20) and (21) will no longer hold because of the corruption of the power control bits. The power control subchannel transmits at a rate of one bit every 1.25 ms. Using the puncturing technique, each power control bit which is two symbols long replaces two consecutive downlink traffic channel code symbols. Since the code symbols have a rate of 19.2 kbps, there will be one power control bit transmitted within every 24 code symbols. There are 16 possible starting positions for the power control bit. Each position corresponds to one of the first 16 code symbols within a 1.25 ms period. In each 1.25 ms period, a total of 24 bits from the long code sequence are used for scrambling. These bits are numbered 0 through 23. The 4-bit binary number with values 0 through 15, formed from scrambling bits 23, 22, 21, and 20, are used to determine the position of the power control bit. Hence, within every 24 code symbols, the last 7 code symbols are not affected by the power control bits. These uncorrupted code symbols include $r_{18+24i}, r_{19+24i}, \dots, r_{24+24i}, i = 0, 1, \dots, 15$. Table 2 outlines the interleaver output uncorrupted by the power control bits. Those symbols in the dotted boxes in Table 2 can be recovered reliably from r and c . Correspondingly, those symbols in the dotted boxes in Table 1 can be recovered reliably from r and c . Code symbols in the dotted boxes in Table 1 can be divided into 7 groups. Each group contains 16 consecutive code symbols. The 7 groups are listed as follows:

49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64
 113 114 115 116 117 118 119 120 121 122 123 124 125 126 127 128
 177 178 179 180 181 182 183 184 185 186 187 188 189 190 191 192
 241 242 243 244 245 246 247 248 249 250 251 252 253 254 255 256
 305 306 307 308 309 310 311 312 313 314 315 316 317 318 319 320
 337 338 339 340 341 342 343 344 345 346 347 348 349 350 351 352
 369 370 371 372 373 374 375 376 377 378 379 380 381 382 383 384

In the following, we will extract those linear equations in (20) and (21) that are only related to the uncorrupted code symbols in r , such linear equations are called reliable linear equations.

(1) When the data rate is 19.2 ksps, we can not obtain any linear equations from (20) since code symbols are not repeated. In (21), there are 8 reliable linear equations. These reliable linear equations are related with code symbols in the 7th group, that is, s_{369} through s_{384} . Since the constraint length of the (2, 1, 8) convolutional encoder is 9, except for the first and the last 8 linear equations, other linear equations in (21) are related to 18 code symbols. From the structure of H , it is easy to find that the first 8 linear equations are related to the first 2, 4, 6, 8, 10, 12, 14, and 16 code symbols respectively. Similarly, the last 8 linear equations are related to the last 2, 4, 6, 8, 10, 12, 14, and 16 code symbols respectively.

(2) When the data rate is 9.6 ksps, there are 56 reliable linear equations in (20), which are related to the code symbols in the 7 groups. There are 4 reliable linear equations in (21), which are related to the code symbols in the 7th group. Hence, there are a total of 60 reliable linear equations in (20) and (21).

(3) When the data rate is 4.8 ksps, there are 86 reliable linear equations in (20) and 2 reliable linear equations in (21), with a total of 88 reliable linear equations in (20) and (21).

(4) When the data rate is 2.4 ksps, there are 98 reliable linear equations in (20) and 1 reliable linear equations in (21), with a total of 99 reliable linear equations in (20) and (21).

Hence, when the data rate is not 19.2 ksps, an eavesdropper can derive 60 or more reliable linear equations from one data frame. The eavesdropper can solve the reliable linear equations corresponding to the three data rates, 9.6 ksps, 4.8 ksps, and 2.4 ksps. If there are no solutions corresponding to any of the three data rates. The eavesdropper determines that the data rate should be 19.2 ksps. If there is a solution corresponding to one of the three data rates. The long code sequence can be computed and the positions of the power control bits can be determined. Also the corruption caused by the power control bits can be removed by the convolution code. The intercepted code symbols are tested against equations (17) and (18). If the test fails, the solution is incorrect. If there is only one solution passing the test, then long code sequence is found out. If more than one solution pass the tests, then use another frame to test every solution, this process continues until a unique solution is found out.

3.4 A Robust Attack

In the above attack, the eavesdropper needs to solve 4 sets of linear equations corresponding to the 4 data rates, 19.2 kbps, 9.6 kbps, 4.8 kbps, and 2.4 kbps. In the situation when more than one set of linear equations have solutions, the eavesdropper has to determine which solution is correct. This process can be avoided if the eavesdropper can construct a set of linear equations that hold for the 4 data rates. In the following, we will investigate reliable linear equations in (20) and (21) that hold for the 4 data rates.

When the data rate is not 19.2 kbps, the code symbols in u are repeated at least one time. As a result u_{2i} and u_{2i+1} , $0 \leq i \leq 192$, must be equal, i.e.,

$$u_{2i} \oplus u_{2i+1} = 0, \quad 0 \leq i \leq 192. \quad (22)$$

When the data rate is 19.2 kbps, the code symbols in u are not repeated. But, from (13), we can get the following equation

$$u_{383} \oplus u_{384} = 0. \quad (23)$$

By (22), it can be concluded that (23) holds for the 4 data rates. From Table 2,

$$u_{383} = s_{192},$$

$$u_{384} = s_{384}.$$

Since r_{192} and r_{384} are not corrupted by the power control bits, by (15),

$$\begin{aligned} u_{383} &= r_{192} \oplus c(t_{192}), \\ u_{384} &= r_{384} \oplus c(t_{384}). \end{aligned}$$

Hence, it follows that

$$c(t_{192}) \oplus c(t_{384}) = r_{192} \oplus r_{384}. \quad (24)$$

Therefore, from every intercepted frame, the eavesdropper can always construct a reliable linear equation which is independent of the data rate associated with the frame. With 42 intercepted frames, the eavesdropper can construct a set of 42 reliable linear equations to solve for the long code sequence. Each frame lasts for 20 ms, 42 frames last for 840 ms, which is less than 1 second. Hence, by eavesdropping the downlink traffic channel 1 second, the eavesdropper can get enough information to recover the long code sequence. After recovering the long code sequence, the eavesdropper can despread the uplink traffic channel.

4 Concluding Remarks

The analysis of this paper demonstrates that IS-95 CDMA provides a lower than expected level of voice privacy. By eavesdropping the transmission on the downlink traffic channel for one second, an eavesdropper can recover the long

code sequence used for voice privacy. The vulnerability of the voice privacy may have effect on the security of the authentication process since the long code mask is generated during the authentication process. As details of the authentication protocol are not publicly available due to export restriction, we will leave this problem for future research.

There are several reasons for the weak voice privacy. First, channel coding and symbol repetition leak information about the long code sequence, which is demonstrated by (12) and (13). Channel coding provides error correction capability by introducing redundancy in the transmitted information, while ciphers remove redundancy in the information. When error-correcting codes and ciphers are used in the same channel, the effects of error-correcting codes on ciphers should be carefully examined. Second, the block interleaver does not distribute power control bits in a frame uniformly. From Table 1, there are many consecutive code symbols which are not affected by the power control bits. Third, most importantly, the long code generator is not a good cipher. From (3), the long code sequence is a linear feedback shift register sequence. Linear feedback shift register sequence may be good for spread spectrum purpose but not good for cryptographic purpose. It had better design the two sequences separately, one used for spreading, and the other for voice privacy. It is difficult to design a sequence for both spread spectrum and voice privacy. There are different requirements for the two applications. Hence, a new cipher that provides strong voice privacy is required.

Last, we would like to emphasize that attacks described in this paper have not gone through any field test and we have no intention of performing any test in the future. To achieve a high level of voice privacy, the Telecommunication Industry Association (TIA) TR-45 has created a process for developing enhanced encryption algorithms for the next generation CDMA systems. The enhancement will use 128-bit private keys. Several algorithms, including an algorithm designed by the authors of the paper, have been submitted to TR-45 Ad Hoc Authentication Group (AHAG) for adoption of encryption standard for the next generation CDMA systems.

References

1. Ajay Dholakia, *Introduction to Convolution Codes and Applications*, Kluwer Academic Publishers, 1994.
2. Rajian Kurupillai, Mahi Dontamsetti, Fil Cosentino, *Wireless PCS*, McGraw-Hill, 1997.
3. TIA/EIA/IS-95-A, Mobile Station-Base Station Compatibility Standard for Dual-Mode Wideband Spread Spectrum Cellular System.

Attacks on Additive Encryption of Redundant Plaintext and Implications on Internet Security

David A. McGrew and Scott R. Fluhrer

Cisco Systems, Inc.
170 West Tasman Drive, San Jose, CA 95134
{mcgrew,sfluhrer}@cisco.com

Abstract. We present and analyze attacks on additive stream ciphers that rely on linear equations that hold with non-trivial probability in plaintexts that are encrypted using distinct keys. These attacks extend Biham's key collision attack and Hellman's time memory tradeoff attack, and can be applied to any additive stream cipher. We define *linear redundancy* to characterize the vulnerability of a plaintext source to these attacks.

We show that an additive stream cipher with an n -bit key has an effective key size of $n - \min(l, \lg M)$ against the key collision attack, and of $2n/3 + \lg(n/3) + \max(n - l, 0)$ against the time memory tradeoff attack, when the attacker knows l linear equations over the plaintext and has M ciphertexts encrypted with M distinct unknown secret keys.

Lastly, we analyze the IP, TCP, and UDP protocols and some typical protocol constructs, and show that they contain significant linear redundancy. We conclude with observations on the use of stream ciphers for Internet security.

1 Introduction

Biham's key collision (KC) attack [5] and Hellman's time-memory tradeoff (TMTO) attack [11] can be adapted to attack additive encryption in the case that many ciphertexts encrypted with distinct keys, whose corresponding plaintexts all obey some known linear relations, are available to the cryptanalyst. Both of these methods use a precomputation stage in which some knowledge of the typical plaintext is used to build a database, followed by an attack stage in which (hopefully many) ciphertexts are analyzed in an attempt to find unknown keys. The computational cost of the precomputation stage can be amortized over many runs of the attack stage, significantly reducing the effective key size of the cipher against these attacks. These attacks rely on the fact that there are linear equations in the plaintext bits that are known to the cryptanalyst. We define the *linear redundancy* of a plaintext source as the to capture this property.

A linearly redundant source may involve linear equations that hold with probabilities that are not close to unity. We present and analyze an adaptation of the KC attack that works in such cases by using error correcting codes.

We analyze the linear redundancy of the Internet Protocol (IP), the Transmission Control Protocol (TCP), and the User Data Protocol (UDP) traffic with stream ciphers. IP is used by the Internet to transport packets between networks [1], while TCP and UDP are the most common higher-level protocols transported by IP. IP, TCP, and UDP packets are known to contain a significant amount of data that is guessable by an adversary (as was pointed out by Bellare [4]). Our analysis extends these observations by showing that these packets contain a large amount of linear redundancy that can be used in cryptanalytic attacks.

The Stream Cipher ESP (SC-ESP) is a specification for the use of those ciphers to provide privacy within the IPSEC framework [13,9]. It describes how to use additive stream ciphers for the encryption of IP packets (if used in tunnel mode) as well as TCP, UDP, or other packets (if transport mode)¹. Below, we derive requirements on SC-ESP that provide protection against the attacks that we develop in this paper. We do not investigate the linear redundancy of other important Internet protocols, such as HTTP or RTP, though such protocols are commonly used with additive encryption in the SSL, TLS, and SSH protocols. However, the techniques that we develop in this paper do apply to their analysis, and we expect that these protocols also contain a significant amount of linear redundancy.

The rest of this paper is organized as follows. Section 1.1 introduces our terminology and assumptions. Section 2 introduces the idea of linear redundancy. Section 3 introduces the key collision attack, shows how it can be applied to attack additive encryption, and analyzes its computational cost and success probability, while Section 3.1 shows how that attack can be modified to deal with linear equations that are probabilistic, rather than deterministic. Section 4 adapts Hellman's time-memory tradeoff to attack additive encryption, and analyzes the resulting algorithm. The IP, TCP, and UDP protocols are analyzed in Section 5, and are shown to contain enough linear redundancy to enable the successful prosecution of the attacks that we derive. Our conclusions are presented in Section 6.

1.1 Terminology and Assumptions

An additive stream cipher is a cipher that encrypts a plaintext by bitwise adding it (modulo two) to a keystream. The keystream is generated pseudorandomly, given a secret key. Mathematically,

$$c_i = p_i \oplus s_i(k), \quad (1)$$

where c_i , p_i and $s_i(k)$ are the i^{th} bit of the ciphertext, plaintext, and the keystream corresponding to the key k . Additive stream ciphers can be defined over any group, and our results can easily be generalized, but below we consider only binary addition for clarity of exposition.

Modern stream ciphers include RC4, SEAL, the Output Feedback (OFB) mode specified by NIST for use with the DES [18] and the counter mode for

¹ See [8,9] for a more detailed description of IPSEC and ESP

block ciphers [16, p.100]. RC4 is widely used to secure HTTP, as it is part of the Secure Sockets Layer (SSL) and Transport Layer Security (TLS) specifications. Other stream ciphers in use include the recently broken A5/1 [2] used in GSM cellular phones, and the cipher E0 in the Bluetooth specification for Wireless LAN Security [3].

We make the conventional assumption that the cryptanalyst can check if a key is correct by trial decryption of a ciphertext followed by a redundancy check of the decrypted plaintext. We also assume that ciphertexts are distributed uniformly at random, which is essentially equivalent to assuming that the cipher is indistinguishable from a truly random source. We also assume that the cryptanalyst has access to many ciphertexts encrypted under many distinct keys, whose corresponding plaintexts originate from a random but redundant source whose mathematical characterization is known to the attacker. We make the implicit assumption that the unknown keys are distinct, which is a good assumption when the number of unknown keys is less than the square root of the total number of keys, from the ‘birthday paradox’.

2 Linear Redundancy

We generalize the idea of known or guessable plaintext attacks by considering attacks on a large number of ciphertexts whose plaintexts were all generated by the same source. We use the information theoretic idea of a plaintext source as a generator of binary strings that chooses strings by a random process that can be characterized by a probability distribution. A source is *redundant* when its probability distribution is not uniform.

To attack an additive cipher, we consider linear equations in terms of the plaintext bits. From Equation 1, it follows that

$$c_i \oplus c_j = (p_i \oplus p_j) \oplus (s_i(k) \oplus s_j(k)). \quad (2)$$

If $p_i \oplus p_j$ is zero (respectively, one), then $c_i \oplus c_j$ will equal $s_i(k) \oplus s_j(k)$ (respectively, will be its opposite). If the same property holds for a large number of plaintext bits, those bits can be used to identify a collision between a secret key set and a known key set. A single linear relation among the plaintext bits of all plaintexts from a source is equivalent to a single bit of known plaintext, for our purposes.

If there are l linear relationships between the plaintext bits, this fact can be represented mathematically as

$$\bigoplus_{i=1}^w L_{ij} p_i = e_j \text{ for all } j : 1 \leq j \leq l, \quad (3)$$

where L_{ij} is an invertible $m \times w$ boolean matrix, and e is an $l \times 1$ boolean vector.

More generally, Equation 3 can hold with some probability not equal to one. The vector $\delta = Lp \oplus e$, which would be the zero vector if the linear equations

held with probability one, has a low hamming weight. We define the set Δ as the set of typical (e.g., most probable) values of δ , such that

$$\sum_{\delta \in \Delta} P(\delta) = 1 - \epsilon, \quad (4)$$

where $P(\delta)$ is the probability that $Lp \oplus e$ will have the value δ , and ϵ is a number less than one. We say that a plaintext source has *linear redundancy* (λ, ϵ) if there exists an L and e as defined above such that $\lambda = 1 - \lg(\#\Delta)/l$.

In the case that each of the linear equations hold with the same probability ϕ , then the expected weight of $\delta = Lp \oplus e$ is ϕl . In this case, the size of the set of typical vectors Δ is well approximated by $\sum_{i=0}^{\phi l} \binom{l}{i} \simeq 2^{lh(\phi)}$, where $h(\phi) = -\phi \lg \phi - (1-\phi) \lg(1-\phi)$ is the binary entropy function². The linear redundancy then reduces to $(1-h(\phi), 1/2)$. In the following, we focus on the practical attacks rather than the theoretical characterization of linear redundancy.

Our attacks can be viewed as decoding unknown keys, and thus matching them to some set of known keys. From this viewpoint, there is a noisy communication channel from an unknown key to the attacker, where the ‘noise’ is a plaintext message. The unknown keys are the source words, the keystream segments are the code words, ciphertexts are the received words. The attacker faces the problem of decoding the received words to a known code word. We call this channel the *cryptanalytic channel*, and it is analogous to the one defined by Siegenthaler in the description of correlation attacks on combination generators [14]. The code used in our attacks is a set of keys that is randomly chosen by the attacker. Obviously, ciphertexts created with unknown keys that are not in the code cannot be properly decoded. Our attacks work by decoding correctly whenever possible, and rely on the ‘birthday paradox’ to ensure that there are keys common to both the random code and the set of unknown keys.

In some cases, attacks using linear redundancy can be significantly improved through the use of traffic analysis, that is, the use of external information about the ciphertexts to establish the value of the vector e . In the case of Internet security, this information includes the length of the encrypted data, the time of creation of the encrypted data, and the position of each ciphertext in the sequence of all ciphertexts.

3 Key Collision Attacks on Additive Encryption

Key collision attacks [5] take advantage of the birthday paradox to reduce the expected work effort of finding secret keys. These attacks use two distinct sets of keys: a set of unknown secret keys, and a set of keys generated by the cryptanalyst. These sets will contain a common element with high probability when the product of the sizes of the sets is close to the size of the set of all keys.

The known-plaintext key collision attack works as follows: the cryptanalyst encrypts the same fixed plaintext with N distinct keys, and stores the resulting

² This approximation uses the tail inequality [6], and is asymptotically exact

ciphertexts along with the keys that generated them. We call the set of ciphertexts the *known key* set. The cryptanalyst then gets a hold of M ciphertexts that are created by encrypting the same plaintext with distinct unknown keys, and looks for collisions, that is, elements with the same keys that are in both sets. When one of the unknown keys is equal to one of the known keys, a collision occurs. With an n -bit key, this will happen with high probability when $MN \geq 2^n$. In practice, many keys may map the same plaintext to the same ciphertext, so the cryptanalyst must check each collision with a trial decryption.

In order to attack additive encryption of linearly redundant plaintext, we define a *hallmark* of a key. This is a binary vector that captures enough information about the key to enable elements of the known key set to be matched to the unknown key set.

Combining Equations 3 and 1 gives

$$\bigoplus_{i=1}^w L_{ij}s_i(k) = e_j \oplus \bigoplus_{i=1}^w L_{ij}c_i \text{ for all } j : 1 \leq j \leq l. \quad (5)$$

The *known key hallmark* v is defined by $v_j(k) = \bigoplus_{i=1}^w L_{ij}s_i(k)$. The *unknown key hallmark* u is defined by $u_j = e_j \oplus \bigoplus_{i=1}^w L_{ij}c_i$. Both v and u are length l binary vectors. In the event that $u = v$, it is (at least relatively) likely that the known key and unknown key are equal.

We now show how to prosecute a KC attack on an additive cipher, given L and e . In the precomputation stage, compute the set $V = \{(v(k), k) : k \in R\}$ of known keys and their hallmarks, where R is a set of N arbitrary distinct keys, and sort the vectors so that their first components are in non-decreasing order. In the attack stage, we are given the set $C = \{c\}$ of ciphertexts, and we want to find as many of the unknown keys as possible. We denote the number of ciphertexts (and thus the number of unknown hallmarks) as M . The attack algorithm follows:

1. Compute the set of unknown keys and hallmarks $U = \{(Lc \oplus e, c) : c \in C\}$, and sort it into non-decreasing order.
2. Find the join $J = \{(x, k, c) : (x, c) \in U, (x, k) \in V\}$, that is, the intersection of the first components of V and U .
3. For each element $(x, k, c) \in J$, do a trial decryption of the ciphertext c using the key k .

The intersection of two sets of bit vectors can be found by sorting each set into non-decreasing order, maintaining a pointer into each set, repeatedly advancing the pointer that points to the smallest element, and outputting the elements when they match [10]. If radix sort [10] is used, then this algorithm is completely paralellizable.

An important property of this attack is that the vector e need not be known during the precomputation stage; it is sufficient to know e during the attack stage. This property lends itself to practical attacks, as there are many cases in which the plaintext at two locations will be linearly related, though the exact

value of the relationship may only be predictable through traffic analysis or other external means. For example, traffic analysis of the IP protocol can readily discern many TCP/IP packets (TCP ACK packets have the distinctive length of 43 bytes), thus revealing the value of the ‘Protocol’ field of the IP packet (See Table 1).

The basic key collision attack requires storage of order $M + N$. The precomputation stage requires N encryptions and $N \lg N$ comparisons and copies (for sorting).

The attack stage requires $M \lg M$ comparisons and copies in the sorting stage. Finding the set J requires $M + N$ comparisons. The attack performs $\#J$ trial decryptions, which is equal to the sum of the number of false hits, which is $MN/2^l$, and the number of true hits, which is $MN/2^n$. The total computation is of order $M \lg M + M + N + MN(1/2^l + 1/2^n)$.

The expected number of true hits found, that is, the number of messages successfully decrypted, is $MN/2^n$. Thus, the order of the expected work w for each successful decryption is given by

$$\begin{aligned} w &= \frac{M \lg M + M + N + MN(1/2^l + 1/2^n)}{MN/2^n} \\ &= \frac{2^n \lg M}{N} + \frac{2^n}{M} + \frac{2^n}{N} + 2^{n-l} + 1. \end{aligned} \quad (6)$$

If $2^n \lg M/N$ is the leading term in Equation (6), we say that the attack is *sort limited*. If $2^n/M$ is the leading term, we say that the attack is *intersection limited*. This can happen when the size of the known key set is large and the size of the unknown key set is small. If 2^{n-l} is the leading term, we say that the attack is *information limited*. This case happens when there are few linear equations in the plaintext. The term 1 can never be the leading term, as this implies that the known and unknown key sets are larger than the set of all keys. This term can be neglected in practice. The expected number of keys that are tried for a given unknown key hallmark is $N/2^l$. When the attack is information limited, this number is large (on average). When then attack is sort limited or intersection limited, it is small (on average).

To make the advantage over exhaustive search explicit, we introduce the *effective key size*, which we define to be the base-two logarithm of the order of the expected work. The effective key size is denoted as η , and is given by

$$\eta = \lg w = n + \lg \left[\frac{1 + \lg M}{N} + \frac{1}{M} + 2^{-l} + 2^{-n} \right]. \quad (7)$$

When the attack is information limited, then

$$\eta \simeq n + \lg 2^{-l} = n - l. \quad (8)$$

This approximation is valid when $l \ll \lg \min M, N$. The linear relationship between effective key size and l is shown in Figure 1. When the attack is intersection limited, then

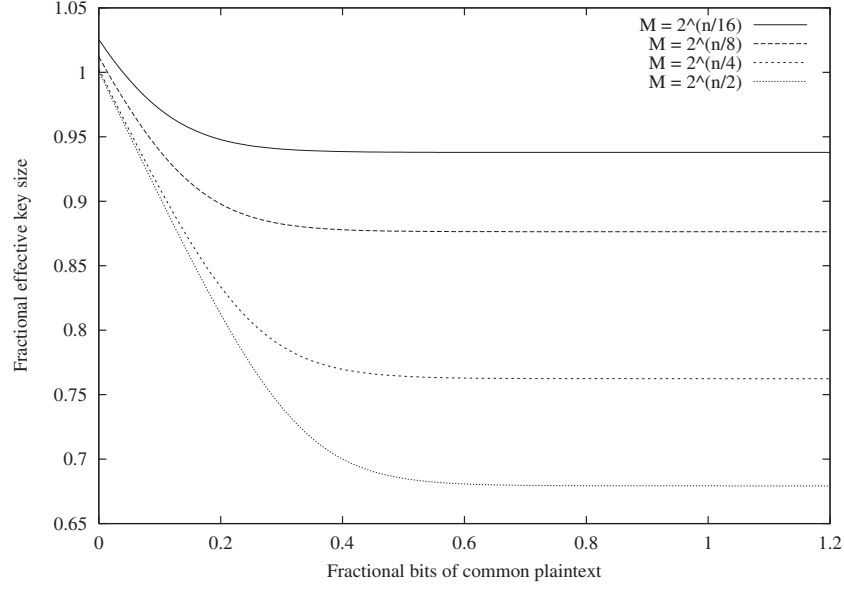


Fig. 1. The effective key size as a function of the linear redundancy. In this figure, the fractional key size η/n is plotted versus the fractional number of linear equations l/n . The plot shows various values of $M/2^n$; in every case, $N = 2^{n/2}$.

$$\eta \simeq n + \lg \left(\frac{1}{M} + \frac{1}{N} \right) \simeq n - \lg \min M, N. \quad (9)$$

When the attack is sort limited, then

$$\eta \simeq n + \lg \frac{\lg M}{N} = n - \lg N + \lg \lg M. \quad (10)$$

We define the *break even value* N_b to be the value of N such that the effective key size is equal to the actual key size; when N is below this value, the attack is not effective. Solving for this value, we find that $N_b = (1 + \lg M)/(1 - 2^{-l} - 2^{-n} - 1/M)$. The effective key size as a function of N can be succinctly expressed as

$$\eta = n - \lg \left[1 + (1 + \lg M) \left(\frac{1}{N} - \frac{1}{N_b} \right) \right]. \quad (11)$$

The dependence of η on N is demonstrated in Figure 2. The maximum possible value for N_b is 2^n , which implies that a necessary condition for the above attack to provide an advantage over exhaustive search is that

$$l \geq \lg \left(1 - \frac{1}{M} - \frac{1 + \lg M}{2^n} \right). \quad (12)$$

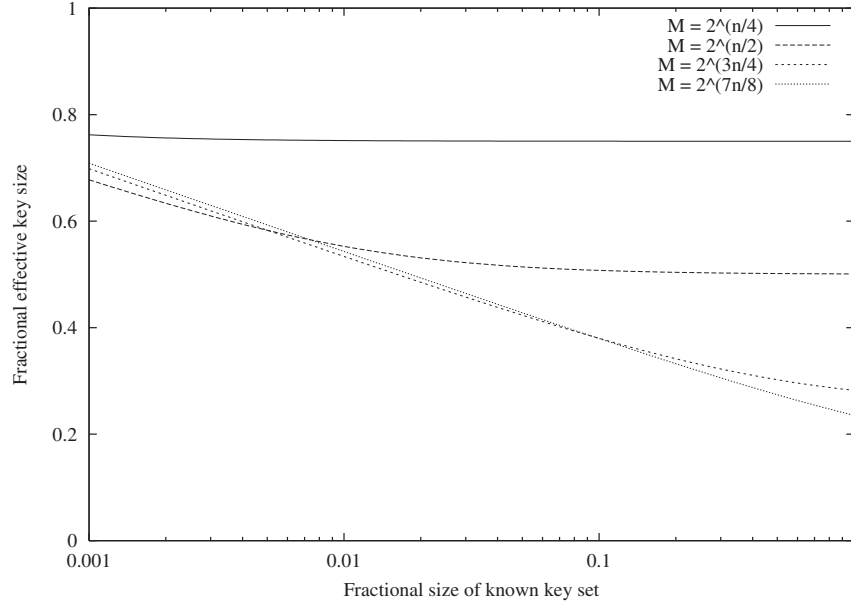


Fig. 2. The effective key size as a function of the size of the unknown key set. In this figure, the fractional key size η/n is plotted versus the fractional size of the known key set $N/2^n$. In these plots, $m = n$

3.1 Probabilistic Linear Equations

The KC attack on additive encryption in the previous section can work even when the linear equations (5) do not always hold, but hold with some non-negligible probability. If the probability that all of the equations hold simultaneously is p , then the effective key size is increased by $\lg 1/p$. However, better attacks can be realized by using error-correcting codes. Below we present a simple adaptation of the KC attack that uses error correction of the hallmarks.

The error-correcting KC attack differs from the KC attack presented above in the precomputation stage and in Step 1. The known key hallmarks are required to be codewords of an error-correcting code D which has codewords of length l , a total of 2^k codewords, and which can correct up to e errors. This property can be realized by using a rejection method during the precomputation stage, which will increase the amount of computation in that stage by a factor of about 2^{l-k} .

Step 1 of the attack algorithm is modified by changing the definition of the set U to $U = \{(d(Lc \oplus e), c) : c \in C\}$, where d is a decoding function for the code D .

The effective key size can be derived as with the information limited case above, with the differences that in the error correcting case the number of false

hits is now $MN/2^k$ and the number of true hits is $pMN/2^n$ ³. The effective key size η_{EC} for the error-correcting KC attack is

$$\eta_{KC} \simeq n - k + \lg \frac{1}{p} = n - lR + \lg \frac{1}{p}, \quad (13)$$

where $R = k/l$ is the rate of the code. There is a tension between R and the decoding error, in that increasing one tends to decrease the other. It is difficult to further characterize the effectiveness of this attack in the general case because of the variety and complexity of error correcting codes [12]. One example of a useful code is the $n = 128, k = 100, e = 4$ code based on BCH codes [15]. Gallager codes [7], which have proved useful in correlation attacks on stream ciphers, may also prove useful in our attacks. It is also possible to use nonlinear codes, though such codes could require a significant storage space.

A strict lower bound on the effective key size of the error correcting KC attack is provided by an information theoretic treatment of the cryptanalytic channel. The capacity C of that channel, which is determined by the plaintext source [6], is the upper bound on the rate R of a code that can be used in the attack, thus limiting that value in Equation (13).

4 Hellman's Time-Memory Tradeoff

Hellman's time-memory tradeoff (TMTO) is a method that can be used to dramatically reduce the average amount of computation needed to invert a one-way function [11]. It works by precomputing a large table, then using the same table to attack many secret keys. Asymptotically, this attack can be used to break a block cipher with an n bit key with about $2^{2n/3}$ operations, using $2^{2n/3}$ storage [11]. Below, we review how to invert a function $S : \mathbb{F}_2^n \rightarrow \mathbb{F}_2^l$, then show how to adapt this result to attack additive encryption of linearly redundant plaintext.

To perform the TMTO, given the function S to be inverted, select a reduction function $R : \mathbb{F}_2^l \rightarrow \mathbb{F}_2^n$, the size of the table N^4 , and the tradeoff parameter t . The reduction function serves to map the range of S back onto its domain. In the precomputation stage, define the function $f(x) = R(S(x))$, and compute the set $T = \{(f^t(x), x) : x \in R\}$, where R is a random N -element subset of \mathbb{F}_2^n , and sort the elements of T so that their first components are in increasing order.

In the attack stage, to find z given y such that $S(z) = y$, compute the set $Y = \{(f^i(R(y)), i) : i = 0, 1, \dots, t-1\}$, and sort its elements so that their first components are in increasing order. For each component $(a, i) \in Y$ such that $(a, x) \in T$ for some x , compute $f^{t-i-1}(x)$ and check if it is the proper inverse.

The precomputation stage requires Nt evaluations of the function f , as well as $N \lg N$ operations for the sorting and storing N elements. The inversion stage

³ This analysis assumes that the decoding function is equally likely to chose any code-word if the number of errors in the hallmark is greater than e , a property which holds for linear codes

⁴ Hellman refers to this parameter as m in [11]. We use the notation N to be consistent with the terminology in the Key Collision section

requires t evaluations of f , sorting and storing t elements, and $N + t$ operations to find the intersection of a t element set and an N element set.

In the TMTO attack against block ciphers, the work effort due to false hits is negligible. R can be chosen so that it does not collide, so that a collision of S implies a collision of the underlying function f . If $l < n$, then this work effort is no longer negligible, as the function f will have more collisions than expected. The expected number of false hits per table look up is bounded by $Nt(t+1)/2^{l+1} \simeq Nt^2/2^{l+1}$.

The success probability of the TMTO attack algorithm is determined by the number σ of elements in the known key set V , where σ can be bounded by,

$$\sigma \leq \sum_{i=1}^N \sum_{j=1}^t \left[\frac{2^n - it}{2^n} \right]^j. \quad (14)$$

Using the choice of parameters $N = t = 2^{n/3}$ suggested in [11], then $\sigma \simeq 2^{2n/3}$. Below, we assume these values.

To use Hellman's time-memory tradeoff in an attack against linearly redundant plaintext encrypted with a stream cipher, f is defined as a mapping from keys to known hallmarks :

$$f(k) = Ls(k) \oplus e. \quad (15)$$

The known key set of hallmarks is the 'logical table' comprised of the iterates of S used in computing T , and the number of distinct elements that it contains is σ .

If the TMTO is done on a set of M unknown hallmarks simultaneously, a set Y must be computed for each unknown hallmark, and the union of the unknown key sets has cardinality tM . In addition, if $n > l$, the time taken to check false hits must be accounted for. The expected work effort w of the TMTO attack is thus for $n \leq l$,

$$\begin{aligned} w &= tM \lg(tM) + tM + N + \frac{tM\sigma}{2^n} \\ &= O(tM \lg(tM) + N). \end{aligned} \quad (16)$$

and for $n > l$,

$$\begin{aligned} w &= tM \lg(tM) + tM + N + \frac{tM\sigma}{2^n} + MNt^3/2^{l+1} \\ &= O(tM \lg(tM) + N + MNt^3/2^l). \end{aligned} \quad (17)$$

The expected number of correct keys that this algorithm finds is $M\sigma/2^n \approx M2^{-n/3}$. Thus the effective key size η_T of the TMTO attack is given by

$$\begin{aligned} \eta_T &= \lg \frac{tM \lg(tM) + N + MNt^3/2^l}{M2^{-n/3}} \\ &= 2n/3 + \lg(n/3 + \lg M + 1/M + 2^{n-l}) \\ &\simeq 2n/3 + \lg n/3 + \max(n-l, 0). \end{aligned} \quad (18)$$

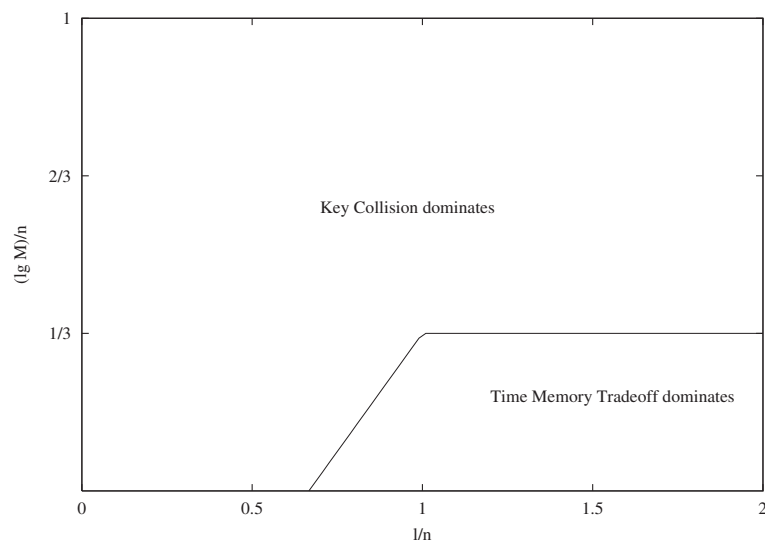


Fig. 3. The ‘phase space’ of attacks on additive encryption, showing which attack dominates as a function of the parameters $\lg M$ and l . Here, the parameters are represented fractionally in terms of the key size n

4.1 Comparison of the TMTO and Key Collision Algorithms

The TMTO attack is more effective than the basic attack when $\eta_T < \eta$. By comparing the estimates for η and η_T given above, we can see that the KC attack is preferable when $\lg(M) > n/3$. The complete ‘phase space’ of attacks on additive encryption is illustrated in Figure 3.

However, the TMTO as described does not work for probabilistic linear equations. In that case, the KC attack has the advantage.

5 Linear Redundancy in IP Packets

We analyzed the IP, TCP, and UDP protocols, and estimated the linear redundancy in the headers of those protocols. A summary of our results is given in Table 1. In this section, all numerals indicate binary expressions.

The Version field is (almost without exception) equal to 0100. The Header Length is nearly always equal to 0101, unless an IP option is used, in which case it is probably 0110. The Precedence/TOS (Type of Service) field is generally set to 00000000. The Protocol field is usually 00000110 (for TCP) or 00001011 (for UDP). The ‘Time to Live’ field is usually 00010000 or less. The ‘Source IP’ and ‘Destination IP’ fields from the IP header are the same in every packet between two particular hosts. Each pair of packets with the same source and destination that can be identified by traffic analysis provides 64 linear equations. The ‘Source Port’ and ‘Destination Port’, in the TCP and UDP protocols, provide a total of

Table 1. Linear redundancy in the headers of the IP, TCP, and UDP protocols. The common values are described in Section 5. The ‘Single Packet’ column shows the redundancy that is detectable in a single packet. The ‘Two Packet’ column shows the redundancy that is present in two consecutive packets from the same source

| Protocol | Field | Size (bits) | Single Packet Redundancy (bits) | Two Packet Redundancy (bits) |
|----------|---------------------|-------------|---------------------------------|------------------------------|
| IP | Version | 4 | 4 | 4 |
| | Header Length | 4 | 4 | 4 |
| | Precedence/TOS | 8 | 8 | 8 |
| | Packet Length | 16 | 4 | 4 |
| | Packet ID | 16 | 0 | 0 |
| | DF bit | 1 | 1 | 1 |
| | MF bit | 1 | 0 | 0 |
| | Fragment Offset | 13 | 0 | 0 |
| | Time to Live | 8 | 3 | 3 |
| | Protocol | 8 | 7 | 7 |
| | Checksum | 16 | 1 | 1 |
| | Source Address | 32 | 0 | 32 |
| | Destination Address | 32 | 0 | 32 |
| | Total | - | 32 | 96 |
| UDP | Source Port | 16 | 0 | 16 |
| | Destination Port | 16 | 0 | 16 |
| | Length | 16 | 0 | 0 |
| | Checksum | 16 | 1 | 1 |
| | Total | - | 1 | 33 |
| TCP | Source Port | 16 | 0 | 16 |
| | Destination Port | 16 | 0 | 16 |
| | Sequence Number | 32 | 0 | 18 |
| | Ack. Number | 32 | 0 | 14 |
| | Data Offset | 4 | 4 | 4 |
| | Checksum | 16 | 1 | 1 |
| | Urgent | 8 | 0 | 0 |
| | Total | - | 5 | 69 |

32 linear equations in the same manner. The TCP ‘Data Offset’ field is usually set to 0101.

5.1 Checksums and Counters

Many protocols use checksums so that transmission errors are likely to be detectable by the receiver. A checksum is an element of a ring, usually \mathbb{F}_2^c or $\mathbb{Z}/2^c$, for some value of c . It is computed by deconcatenating the data into elements of that ring, then summing them together. Checksums over \mathbb{F}_2^c are conventional when the protocol is implemented in hardware, while checksums over $\mathbb{Z}/2^c$ are commonly implemented in software (and are used for IP, TCP, and UDP with $c = 16$).

A checksum over \mathbb{F}_2^c (or CRC) provides c linear equations that always hold. The Bluetooth specification for wireless networking is one example of a protocol that includes such a checksum on data that is encrypted by an additive cipher [3]. A checksum over $\mathbb{Z}/2^n$ provides one linear equation that always holds, since the least significant bit of a sum of integers is equal to the exclusive or of the least significant bits of the integers. Probabilistic linear equations in other bits of the checksum can be derived, but will be poor approximations if the number of integers summed together is large.

In many protocols, an integer called a *counter* is included in each packet, and is used to indicate the ordering of the packets to the receiver. Counters may be incremented by one for each new packet, or may be incremented by some other value (e.g., the number of bytes contained in the data portion of the packet, as is done in the TCP protocol). If a c -bit counter x appears in a packet, and $x + y$ appears in another packet, where $y < 2^q$, for some q , then

$$x_{i+q} = (x + y)_{i+q} \text{ with probability } \geq 1 - 2^{-i}. \quad (19)$$

A c bit counter that increments by a value less than 2^c provides a significant amount of information.

IP, TCP, and UDP all use checksums over $\mathbb{Z}/2^{16}$. The low bit of the checksum is a linear function of the other packet data, from Section 5.1. If the layer three protocol of a packet is known, then the checksums provide two linear equations that hold with probability one.

TCP packets contain a 32-bit counter that is incremented by the length (in bytes) of the packet's data. These lengths will be no more than 1500 (which is the Ethernet MTU) with high probability. Since $2^{11} > 1500$, two adjacent counters provide 18 linear equations that hold with probability $7/8$ or greater. To use these equations in an attack requires some traffic analysis to discover two sequential TCP packets. The TCP 'Acknowledgement Number' similarly provides about 14 linear equations.

6 Conclusions

Practical attacks on additive stream ciphers that rely on linear equations over the plaintext bits are possible, even when those equations hold probabilistically. The IP, TCP, and UDP protocol headers have a significant amount of linear redundancy, and are vulnerable to these attacks. In practice, effective key sizes of Internet encryption are close to $n - \lg M$, when a cryptanalyst has M ciphertexts encrypted under distinct keys available. We conjecture that only protocols specifically designed to not be linearly redundant will not be vulnerable to these attacks. Compression would reduce the linear redundancy of a source; however, we are pessimistic about the effectiveness of using compression to protect against our attacks in practice.

While our attacks are powerful, there is an easy defense against them: increase the key size of the cipher. Cipher keys can be extended in ways that are not secure against other forms of attack (e.g., 'whitening' with a fixed value) and

still provide resistance to our attacks. This approach is similar to the idea of concatenating ‘salt’ (e.g., unique but public data) to a secret password in order to reduce the effectiveness of attacks that amortize effort across many passwords. variable size key, although in common usage its key size is 128 bits.

The attacks that we outline are possible against Internet traffic encrypted with 128-bit RC4 with a complexity of about 2^{88} , assuming that an adversary can intercept ciphertexts from 2^{40} distinct sessions. This number is feasible; a single Internet site that establishes 2^{32} SSL connections per day has been reported [17]. While this attack is beyond the limit of current cryptanalytic technology, it is worth noting that it does no harm to increase the key size to compensate for our attacks: the throughput of the RC4 cipher is independent of its key size.

The attacks that we presented rely on the fact that the secret keys are chosen uniformly at random. If the keys are chosen from a highly skewed probability distribution (e.g., a broken random number generator that outputs the same number every time), the effectiveness of our attacks is significantly reduced. Of course, the broken random number generator creates other security problems!

Considerable future work remains untouched. While we established the viability of attacks relying on the redundancy of plaintext encrypted by additive stream ciphers, we did not investigate efficient decoding methods for use when the linear equations are probabilistic. Also, it may be possible to extend the time-memory tradeoff approach so that it can be used in the probabilistic case.

References

1. Postel, J., “The Internet Protocol”, IETF RFC 791, USC/Information Sciences Institute, September 1981.
2. Briceno, M., Goldberg, I., Wagner, D., “A pedagogical implementation of A5/1”, <http://www.scard.org>, May 1999.
3. Bluetooth SIG, “BLUETOOTH Baseband Specification Version 1.0 B, Section 14.3”, <http://www.bluetooth.com>.
4. Bellovin, S., “Probable Plaintext Cryptanalysis of the IP Security Protocols,” Proceedings of the Symposium on Network and Distributed System Security, pp. 155-160, 1997.
5. Biham, E., “How to Forge DES-Encrypted Messages in 2^{28} Steps”, Technion Computer Science Department Technical Report CS0884, 1996.
6. Blahut, R., “Principles and Practice of Information Theory”, Addison-Wesley, 1983.
7. Gallager, R., “Low Density Parity Check Codes”, IEEE Transactions on Information Theory, IT-8 pp. 21-28, January, 1962.
8. Kent, S., and R. Atkinson, “Security Architecture for IP”, RFC 2401, November 1998.
9. Kent, S., and R. Atkinson, “IP Encapsulating Security Payload (ESP)”, RFC 2406, November 1998.
10. Knuth, D., “The Art of Computer Programming: Volume Three, Sorting and Searching”, Addison-Wesley, 1998.
11. Hellman, M. E., “A cryptanalytic time-memory trade-off”, IEEE Transactions on Information Theory, July 1980, pp. 401-406.

12. Litsyn, S., Rains, E.M., Sloane, N.J., "Table of Nonlinear Binary Codes", <http://www.research.att.com/~njas/codes/And>.
13. McGrew, D., Fluhrer, S., "The Stream Cipher Encapsulating Security Payload," draft-mcgrew-ipsec-scesp-01.txt, Internet Draft, July, 2000.
14. Siegenthaler, T., "Correlation-immunity of nonlinear combining functions for cryptographic applications," IEEE Transactions on Information Theory, Vol. IT-30, pp. 776-780, October, 1984.
15. Sloane, N. J, Reddy, S., and Chen., C., "New binary codes". IEEE Transactions on Information Theory, IT-18, pp. 503-510, 1972.
16. Schneier, B., "Applied Cryptography", New York: Wiley, 1996.
17. van Someren, N., "There will be no cryptographic abundance without cryptographic hardware", Xerox PARC Symposium on Life in an Era of Cryptographic Abundance, June, 2000.
18. U. S. National Institute of Standards and Technology, "DES Modes of Operation", Federal Information Processing Standards Publication 81, 1980.

Cryptanalysis of the “Augmented Family of Cryptographic Parity Circuits” Proposed at ISW’97

A.M. Youssef

Center for Applied Cryptographic Research, Department of Combinatorics and Optimization, University of Waterloo, Waterloo, ON N2L 3G1
a2youssef@cacr.math.uwaterloo.ca

Abstract. At Crypto’90, Koyama and Terada proposed a family of cryptographic functions for application to symmetric block ciphers. Youssef and Tavares showed that this family is affine and hence it is completely insecure. In response to this, Koyama and Terada modified their design, by including a data dependent operation between layers. The modified family of circuits was presented in the first international security workshop (ISW’97). In this paper, we show that the modified circuit can be easily broken by a differential-like attack. More explicitly, we show that after d rounds, and for any specific key K , the input space can be partitioned into $M \leq 2^d$ sets such that the ciphertext Y of each set is related to the plaintext X by an affine relation. The expected value of $M \ll 2^d$. Our attack enables us to explicitly recover these linear relations. We were able to break an 8-round 64-bit version of this family in few minutes on a workstation using less than 2^{20} chosen plaintext-ciphertext pairs.

Keywords: Block cipher, cryptanalysis, augmented parity circuits

1 Introduction and Definitions

Koyama and Terada [2] proposed a family of cryptographic functions called “non-linear” parity circuits. Youssef and Tavares [7] showed that this family of functions is affine over $GF(2)$ and hence it is completely insecure. In [3], Koyama and Terada introduced a random involution called Value-Dependent-Swapping (VDS). In the VDS, the left half and the right half of a sequence of bits are swapped if its parity is odd. In [4],[5] the VDS was incorporated into DES in order to make it stronger against differential and linear cryptanalysis. By including this VDS in the parity circuits proposed in [2], Koyama and Terada obtained what they called an augmented version of their cryptographic functions family. The following definitions are given in [3].

Definition 1. Let $x = L||R$ be a sequence of $2k$, $k > 0$ bits where L stands for left half of x and R stands for right, $length(L) = length(R) = k$. A value dependent swapping, or $V(x)$, is defined to be

$$V(x) = \begin{cases} R||L & \text{if } h(x) = 0, \\ L||R & \text{if } h(x) = 1, \end{cases} \quad (1)$$

where $h(x) \in \{0, 1\}$.

Definition 2. Let $x = x_l||x_r$ be a sequence of $2k$, $k > 0$ bits where x_l stands for left half of x and x_r stands for right, $\text{length}(x_l) = \text{length}(x_r) = k$. A VDS, which is an involution value-dependent-swapping based on the parity of the weight of x , is defined to be

$$V(x) = \begin{cases} x_r||x_l & \text{if } \text{weight}(x) \text{ is odd,} \\ x_l||x_r & \text{if } \text{weight}(x) \text{ is even,} \end{cases} \quad (2)$$

where $\text{weight}(x)$ is the number of 1's in the bit sequence x .

Definition 3. A parity layer with length n , or simply an $L(n)$ circuit layer, is a Boolean device with an n -bit input and n -bit output, characterized by a key that is a sequence of n symbols from $\{0, 1, +, -\}$.

Definition 4. A function $B = f(K, A)$ computed by an $L(n)$ circuit layer with key $K = k_1k_2 \cdots k_n \in \{0, 1, +, -\}^n$ is the relation from an n -bit input sequence $A = a_1a_2 \cdots a_n \in \{0, 1\}^n$ to an n -bit sequence $B = b_1b_2 \cdots b_n \in \{0, 1\}^n$ defined below. An $L(n)$ circuit layer computes first the variable T modulo 2 such that

$$T = \bigoplus_{j=1}^n t_j, \quad (3)$$

where

$$t_j = \begin{cases} 1 & \text{if } (k_j = 0 \text{ and } a_j = 0) \text{ or } (k_j = 1 \text{ and } a_j = 1), \\ 0 & \text{Otherwise.} \end{cases} \quad (4)$$

The output $B = b_1b_2 \cdots b_n$ of the circuit layer is then

$$b_j = \begin{cases} \begin{cases} k_j = - \text{ and } T = 1 \\ \text{or} \\ k_j = + \text{ and } T = 0 \\ \text{or} \\ k_j = 1 \end{cases} & \text{if } \begin{cases} k_j = - \text{ and } T = 1 \\ \text{or} \\ k_j = + \text{ and } T = 0 \\ \text{or} \\ k_j = 1 \end{cases} \\ \overline{a_j} & \text{if } \begin{cases} k_j = - \text{ and } T = 1 \\ \text{or} \\ k_j = + \text{ and } T = 0 \\ \text{or} \\ k_j = 1 \end{cases} \\ a_j & \text{Otherwise.} \end{cases} \quad (5)$$

Definition 5. A parity circuit of width n and depth d , or simply $C(n, d)$ circuit, is a matrix of d $L(n)$ circuit layers with keys denoted by $K = K_1||K_2 \cdots K_d$ for which the n output bits of the $(i-1)$ -th circuit layer are the n input bits for the i -th circuit layer, for $2 \leq i \leq d$. The key for the $C(n, d)$ circuit is a $d \times n$ matrix with its d lines containing circuit layer keys.

Table 1. $C_+(n, d)$ with $n = 10$ and $d = 3$

| <i>Input</i> | 1 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | Swap |
|---------------|---|---|---|---|---|---|---|---|---|---|------|
| K_1 | - | 0 | 1 | - | + | + | 1 | 1 | - | + | |
| <i>Output</i> | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | yes |
| K_2 | + | 1 | 0 | 1 | 1 | + | 0 | - | + | - | |
| <i>Output</i> | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | no |
| K_3 | - | 0 | 1 | + | + | 0 | - | + | + | - | |
| <i>Output</i> | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 1 | 1 | yes |

Let F be the function from $\{0, 1\}^n$ to $\{0, 1\}^n$ computed by a circuit $C(n, d)$ with key $= K || K_2 \cdots K_d$. That is $F(K, A)$ is defined as

$$F(K, A) = f(K_d, f(K_{d-1}, \dots, f(K_1, A) \cdots)). \quad (6)$$

By showing that, for any fixed key, the $C(n, d)$ circuit can be constructed using XOR gates only, Youssef and Tavares [7] showed that $F(K, A)$ above is affine over $GF(2)$.

Definition 6. A function $B = f_+(K, A)$ computed by an augmented $L(n)$ circuit layer with key K , or simply $L_+(n)$ layer, is the function $V(f(K, A))$, where V is the VDS function as in Definition 2, and f is the function computed by an $L(n)$ circuit layer.

Definition 7. A augmented parity circuit of width n and depth d , or simply $C_+(n, d)$ circuit, is a matrix of d $L_+(n)$ circuit layers with keys denoted by $K = K_1 || K_2 \cdots K_d$ for which the n output bits of the $(i - 1)$ -th circuit layer are the n input bits for the i -th circuit layer, for $2 \leq i \leq d$. The key for the $C_+(n, d)$ circuit is a $d \times n$ matrix with its d lines containing circuit layer keys. A F_+ function from $\{0, 1\}^n$ to $\{0, 1\}^n$ computed by a circuit $C(n, d)$ with key $= K || K_2 \cdots K_d$ as

$$F_+(K, A) = f_+(K_d, f_+(K_{d-1}, \dots, f_+(K_1, A) \cdots)). \quad (7)$$

Table 1 shows the example given in [3] for a $C_+(n, d)$ circuit with $n = 10$ and $d = 3$

2 Cryptanalysis of the $C_+(n, d)$ Circuit

Since the $C(n, d)$ circuit is affine [7], the $C_+(n, d)$ circuit can be viewed as a composition of key-dependent affine transformations and the VDS layer (see Figure 1). Thus the security of the $C_+(n, d)$ relies heavily on the cryptographic strength of the VDS layer.

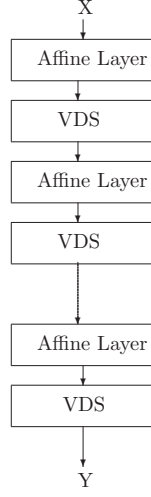


Fig. 1. The $C_+(n, d)$ viewed as a composition of affine and VDS layers

Observation 01 For any specific key k , the ciphertext Y of the $C_+(n, d)$ circuit is related to the plaintext X by one of the affine relations

$$Y = A_i(k)X \oplus b_i(k), \quad (8)$$

where $i = 1, 2, \dots, M$, $A_i(k)$ is a key-dependent non singular binary matrix, $b_i(k)$ is a key-dependent $n \times 1$ binary vector and $M \leq 2^d$.

Proof. Let VDS_i denote the swap variable at round i . I.e., $VDS_i = 0$ if the parity of the input to the VDS layer at round d is even and $VDS_i = 1$ if this parity is odd. Thus $VDS_i \in \{0, 1\}$ and hence for a $C_+(n, d)$ circuit, $VDS_1, \dots, VDS_d \in \{0, 1\}^d$. Thus the input space of the $C_+(n, d)$ circuit can be partitioned into 2^d sets

$$S_1, S_2, \dots, S_{2^d}, \quad (9)$$

where for any fixed $1 \leq i \leq 2^d$, VDS_1, \dots, VDS_d is fixed and hence the d VDS layers can be modeled by fixed bit permutation layers. The output Y corresponding to the input $X \in S_i$ can be obtained by a composition of fixed affine relations and hence Y is related to X by a fixed affine relation for all $X \in S_i$. Since there is no guarantee that all the 2^d possible values of VDS_1, \dots, VDS_{2^d} will appear, then $M \leq 2^d$. \square

Figure 2 illustrates the $C_+(n, d)$ equivalent circuit according to observation 01 above. The following observation illustrates how the “swap control” function in this figure operates. By noting that VDS_i is a linear function of the input to layer i , then we have

Observation 02 Inputs that belong to the same set in observation 01 above must satisfy a set of d linear equations.

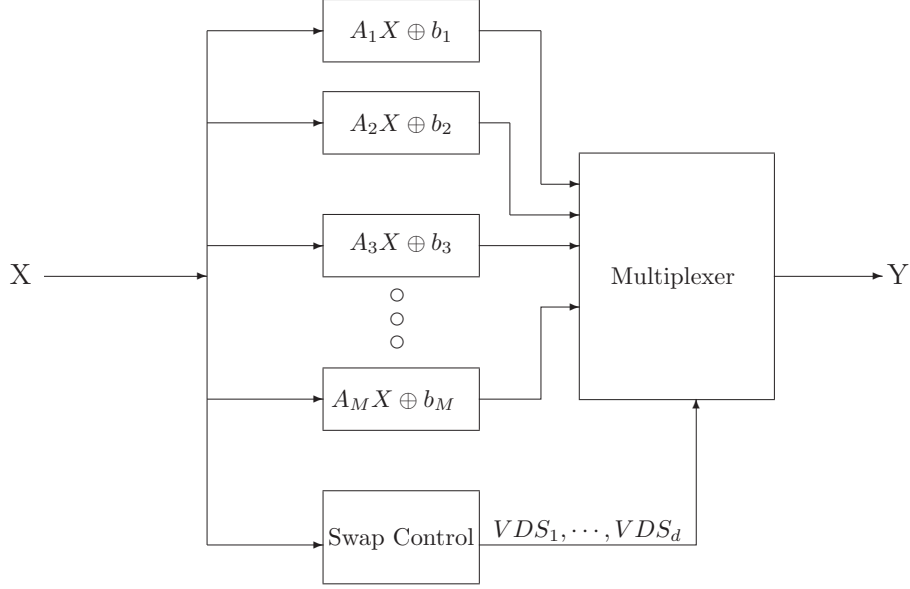


Fig. 2. Equivalent circuit of the $C_+(n, d)$ according to observation 01

For a given known key, these 2^d (d linear relations) can be derived by calculating the parity of the input to the d VDS layers in terms of the input X . If some of these linear relations don't have a solution, then M will be less than 2^d . Figure 4 shows the linear relations corresponding to Example 1 in [3]. Note that for this particular example, we have more than one possible solution for A_i s and b_i s. Figure 4 shows only one of these possible solutions. While observations 01 and 02 are enough to cause uneasy feeling when using the $C_+(n, d)$ for most practical values of d , we extend our attack to find these linear relations. The main idea is to develop an algorithm that can be used to group the input/output pairs that belong to the same set S_i and then solve a set of linear equations to find the Matrix A_i and the vector b_i . The attack makes use of the following observation

Observation 03 For the $C_+(n, d)$, if the input R_1, R_2 and R_3 belong to the set S_i , then

$$R_4 = R_3 \oplus (R_1 \oplus R_2)$$

belongs to the same set S_i .

Proof. If R_1, R_2 and $R_3 \in S_i$ then they must satisfy a set of d linear equations in the form

$$CR_1 = b, CR_2 = b, CR_3 = b,$$

where C is an $d \times n$ matrix and b is a $d \times 1$ vector. The observation is proved by noting that

$$CR_4 = CR_3 \oplus CR_2 \oplus CR_1 = b$$

```

1.  $R_1 = \text{Random}()$ 
2. do
3. {
4.  $pass = 0$ 
5.  $R_2 = \text{Random}()$ 
6.  $\delta_x = R_1 \oplus R_2$ 
7. for  $i = 1$  to  $i = Trials$ 
8. {
9.  $R_3 = \text{Random}()$ 
10.  $R_4 = R_3 \oplus \delta_x$ 
11.  $\delta_y = F_+(R_1) \oplus F_+(R_2) \oplus F_+(R_3) \oplus F_+(R_4)$ 
12. if  $(\delta_y = 0)$  increment  $pass$ 
13. }
14. if  $(pass \geq Threshold)$  Declare  $R_1$  and  $R_2 \in$  same set
15. }while number of collected pairs  $\leq P$ 

```

Fig. 3. Basic steps in the attack

and hence R_4 also satisfy this set of equation. Thus R_4 must belong to the same set S_i . \square

Note that if R_1, R_2 , and $R_3 \in S_i$ then for any key K

$$F_+(K, R_1) \oplus F_+(K, R_2) \oplus F_+(K, R_3) \oplus F_+(K, (R_3 \oplus (R_1 \oplus R_2))) = 0 \quad (10)$$

In our attack, we pick random triples R_1, R_2 and R_3 and test for the condition in equation (10). Since there is no guarantee that R_3 will belong to S_i even if R_1 and R_2 do, we repeat the test for different values of R_3 ($Trials$ in Figure 3). We decide that R_1 and R_2 are in the same set if the condition is satisfied for a large number of times ($Threshold$ in Figure 3). Wrong decisions by the algorithm (i.e., if the algorithm declares that R_2 and R_1 are in the same set while they are not) can be filtered out by collecting more than $n + 1$ pairs (e.g., $P = 2n$ pairs) because with high probability the resulting set of equations we will try to solve will be inconsistent if the algorithm accepts wrong pairs. Another method to prevent the algorithm from accepting wrong pairs is to increase the value of $Trials$ and make the value of $Threshold$ very close to $Trials$. However, this may increase the number of plaintext-ciphertext pairs required to break the algorithm. Throughout these experiments, the value of $Threshold$ was set based on the statistics of the $pass$ variable (see Figure 3). We set $Threshold$ close to the maximum value of $pass$.

3 Analysis of the Algorithm and Experimental Results

Assuming that the size of the input sets are equal, then the probability that R_1, R_2 and R_3 are in the same set is $\frac{1}{M^3}$ where M is the number of partitions.

Table 2. Average number of sets versus optimal value for $n = 10$

| d | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 |
|-----------------|---|---|---|----|----|----|-----|-----|-----|------|------|------|------|------|------|------|
| $Average(M)$ | 2 | 3 | 4 | 7 | 11 | 15 | 25 | 37 | 57 | 62 | 100 | 143 | 162 | 232 | 325 | 393 |
| $min(2^d, 2^n)$ | 2 | 4 | 8 | 16 | 32 | 64 | 128 | 256 | 512 | 1024 | 1024 | 1024 | 1024 | 1024 | 1024 | 1024 |

The maximum value for M is $min(2^d, 2^n)$. Thus the number of chosen plaintext-ciphertext pairs required for the attack increases with M^3 . In other words, the success of the attack depends heavily on the number of the input partitions. The intensive use of bit oriented operations in the $C_+(n, d)$ circuits puts an upper-bound on d , and consequently M , for any efficient software implementation. The average number of partitions for $n = 10$ is shown in Table 2. Each point represents an average over 100 $C_+(n, d)$ circuits with randomly selected keys. It is clear that this number is much less than the optimum value $max(2^d, 2^n)$. Our experimental results shows that this large deviation from the optimum case holds for larger block lengths. It is also easy to prove that if the key K is restricted to the set $\{0, 1\}$ instead of $\{0, 1, +, -\}$, then $M \leq 2$ for all $d \geq 1$. Note that because we don't know M in advance, it is hard to optimize the choice of *Trials* and *Threshold* to minimize the number of plaintext-ciphertext pairs required for the attack. Moreover, our experiments shows that the $C_+(n, d)$ circuit fails to behave like a random function for practical values of d and hence it is not easy to predict the probability of wrong pairs satisfying equation (10) based on the random function model. The good point (from the attacker point of view) is that the attack works almost all the time. In many cases, we were able to break an 8-round 64-bit version of this family in few minutes on a workstation using less than 2^{20} chosen plaintext-ciphertext pairs.

Remark 1. The non-affineness defined in [3] doesn't provide a useful measure of resistance against linear attacks. The nonlinearity of a function f is defined as the minimum distance between the set of affine functions and all the non-zero linear combinations of the output coordinates of f [6]. Our experiments shows that for practical values of d , the average nonlinearity of the $C_+(n, d)$ circuits is very poor compared to the expected nonlinearity of randomly selected functions of the same size n . Thus it is conceivable that the $C_+(n, d)$ circuit be broken using a variant of linear cryptanalysis [6].

4 Conclusion

The security of the $C_+(n, d)$ circuit relies only on the cryptographic strength of the VDS function because the rest of the circuit is affine. Controlling the swapping based on the parity results in a cryptographically weak function. Thus for practical values of n and d , the augmented family of parity circuits $C_+(n, d)$ proposed by Koyama and Terada is insecure.

$$\begin{array}{l}
\text{if } \begin{bmatrix} 1111111111 \\ 1000010111 \\ 1111100011 \end{bmatrix} X = \begin{bmatrix} 0 \\ 1 \\ 1 \end{bmatrix} \text{ then } Y = \begin{bmatrix} 0001110100 \\ 0111111100 \\ 0000011000 \\ 0110010010 \\ 0110010001 \\ 1110010000 \\ 0100000000 \\ 0010000000 \\ 0000111000 \\ 0001011000 \end{bmatrix} X \oplus \begin{bmatrix} 0 \\ 1 \\ 0 \\ 1 \\ 1 \\ 1 \\ 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}, \\
\\
\text{if } \begin{bmatrix} 1111111111 \\ 1011110000 \\ 0001111111 \end{bmatrix} X = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} \text{ then } Y = \begin{bmatrix} 1010000011 \\ 1110001111 \\ 1100000000 \\ 1001001100 \\ 1000101100 \\ 1000011100 \\ 0000001000 \\ 0000000100 \\ 1100000001 \\ 1100000010 \end{bmatrix} X \oplus \begin{bmatrix} 1 \\ 0 \\ 1 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}, \\
\\
\text{if } \begin{bmatrix} 1111111111 \\ 1000010111 \\ 0110010111 \end{bmatrix} X = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \text{ then } Y = \begin{bmatrix} 1010000011 \\ 1110001111 \\ 1100000000 \\ 1001001100 \\ 1000101100 \\ 1000011100 \\ 0000001000 \\ 0000000100 \\ 1100000001 \\ 1100000010 \end{bmatrix} X \oplus \begin{bmatrix} 1 \\ 0 \\ 1 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}, \\
\\
\text{if } \begin{bmatrix} 1111111111 \\ 1011110000 \\ 1011101100 \end{bmatrix} X = \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix} \text{ then } Y = \begin{bmatrix} 1010000011 \\ 1110001111 \\ 1100000000 \\ 1001001100 \\ 1000101100 \\ 1000011100 \\ 0000001000 \\ 0000000100 \\ 1100000001 \\ 1100000010 \end{bmatrix} X \oplus \begin{bmatrix} 1 \\ 0 \\ 1 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}.
\end{array}$$

Fig. 4. (continued)

References

1. Eli Biham and Adi Shamir, *Differential cryptanalysis of DES-like cryptosystems*, Journal of Cryptology, Vol. 4, No. 1, pp. 3-72, 1991.
2. K. Koyama and R. Terada, *Nonlinear Parity Circuits and their cryptographic applications*, Advances in Cryptology, Proceedings of Crypto'90, LNCS537, pp. 582-599, Springer-Verlag, 1991.
3. K. Koyama and R. Terada, *An Augmented Family of Cryptographic Parity Circuits*, Proceeding of Information Security Workshop (ISW'97), LNCS1396, pp.198-208, Springer-Verlag, 1998.
4. T. Kaneko, K. Koyama and R. Terada, *Dynamic swapping schemes and differential cryptanalysis* IEICE Transactions on Fundamentals, vol. E77-A, pp. 1328-1336, 1994.
5. Y. Nakao, K. Koyama and R. Terada, *The security of an RDES cryptosystem against linear cryptanalysis* IEICE Transactions on Fundamentals, vol. E79-A, pp. 12-19, 1996.
6. M. Matsui, *Linear Cryptanalysis method for DES cipher* Advances in Cryptology, Proceedings of Eurocrypt'93, LNCS 765, pp. 386-397, Springer-Verlag, 1994.
7. A.M. Youssef and S.E. Tavares, *Cryptanalysis of the "Non-linear parity circuits" proposed at crypto'90*, IEE Electronics Letters, Vol.33, No. 7, pp. 585-586, 1997.

Camellia: A 128-Bit Block Cipher Suitable for Multiple Platforms – Design and Analysis

Kazumaro Aoki¹, Tetsuya Ichikawa², Masayuki Kanda¹, Mitsuru Matsui²,
Shiho Moriai¹, Junko Nakajima², and Toshio Tokita²

¹ Nippon Telegraph and Telephone Corporation,
1-1 Hikarinooka, Yokosuka, Kanagawa, 239-0847 Japan
{maro,kanda,shiho}@isl.ntt.co.jp

² Mitsubishi Electric Corporation,
5-1-1 Ofuna, Kamakura, Kanagawa, 247-8501 Japan
{ichikawa,matsui,june15,tokita}@iss.isl.melco.co.jp

Abstract. We present a new 128-bit block cipher called *Camellia*. Camellia supports 128-bit block size and 128-, 192-, and 256-bit keys, i.e., the same interface specifications as the Advanced Encryption Standard (AES). Efficiency on both software and hardware platforms is a remarkable characteristic of Camellia in addition to its high level of security. It is confirmed that Camellia provides strong security against differential and linear cryptanalyses. Compared to the AES finalists, i.e., MARS, RC6, Rijndael, Serpent, and Twofish, Camellia offers at least comparable encryption speed in software and hardware. An optimized implementation of Camellia in assembly language can encrypt on a Pentium III (800MHz) at the rate of more than 276 Mbits per second, which is much faster than the speed of an optimized DES implementation. In addition, a distinguishing feature is its small hardware design. The hardware design, which includes encryption and decryption and key schedule, occupies approximately 11K gates, which is the smallest among all existing 128-bit block ciphers as far as we know.

1 Introduction

This paper presents a 128-bit block cipher called *Camellia*, which was jointly developed by NTT and Mitsubishi Electric Corporation. Camellia supports 128-bit block size and 128-, 192-, and 256-bit key lengths, and so offers the same interface specifications as the Advanced Encryption Standard (AES). The design goals of Camellia are as follows.

High Level of Security. The recent advances in cryptanalytic techniques are remarkable. A quantitative evaluation of security against powerful cryptanalytic techniques such as differential cryptanalysis [4] and linear cryptanalysis [18] is considered to be essential in designing any new block cipher. We evaluated the security of Camellia by utilizing state-of-art cryptanalytic techniques. We have confirmed that Camellia has no differential and linear characteristics that hold

with probability more than 2^{-128} . Moreover, Camellia was designed to offer security against other advanced cryptanalytic attacks including higher order differential attacks [13,10], interpolation attacks [10,2], related-key attacks [5,15], truncated differential attacks [13,23], boomerang attacks [26], and slide attacks [6,7].

Efficiency on Multiple Platforms. As cryptographic systems are needed in various applications, encryption algorithms that can be implemented efficiently on a wide range of platforms are desirable, however, few 128-bit block ciphers are suitable for both software and hardware implementation. Camellia was designed to offer excellent efficiency in hardware and software implementations, including gate count for hardware design, memory requirements in smart card implementations, as well as performance on multiple platforms.

Camellia consists of only 8-by-8-bit substitution tables (*s*-boxes) and logical operations that can be efficiently implemented on a wide variety of platforms. Therefore, it can be implemented efficiently in software, including the 8-bit processors used in low-end smart cards, 32-bit processors widely used in PCs, and 64-bit processors. Camellia doesn't use 32-bit integer additions and multiplications, which are extensively used in some software-oriented 128-bit block ciphers. Such operations perform well on platforms providing a high degree of support, e.g., Pentium II/III or Athlon, but not as well on others. These operations can cause a longer critical path and larger hardware implementation requirements.

The *s*-boxes of Camellia are designed to minimize hardware size. The four *s*-boxes are affine equivalent to the inversion function in the finite field $\text{GF}(2^8)$. Moreover, we reduced the inversion function in $\text{GF}(2^8)$ to a few $\text{GF}(2^4)$ arithmetic operations. It enabled us to implement the *s*-boxes by fewer gate counts.

The key schedule is simple and shares part of its procedure with encryption. It supports on-the-key subkey generation and subkeys are computable in any order. The memory requirement for generating subkeys is quite small; an efficient implementation requires about 32-byte RAM for 128-bit keys and about 64-byte RAM for 192- and 256-bit keys.

Outline of the Paper. This paper is organized as follows: Sect. 2 describes the notations and high-level structure of Camellia. Section 3 defines each components of the cipher. Section 4 describes the rationale behind Camellia's design. In Sect. 5 we evaluate Camellia's strength against known attacks. Section 6 contains the performance of Camellia. We conclude in Sect. 7.

2 Structure of Camellia

Camellia uses an 18-round Feistel structure for 128-bit keys, and a 24-round Feistel structure for 192- and 256-bit keys, with additional input/output whitenings and logical functions called the *FL*-function and *FL*⁻¹-function inserted every 6 rounds. Figures 1 shows an overview of encryption using 128-bit keys. An element with the suffix _(*n*) shows that the element is *n*-bit long.

The key schedule generates 64-bit subkeys kw_t ($t = 1, 2, 3, 4$) for input/output whitenings, k_u ($u = 1, 2, \dots, r$) for round functions and kl_v ($v = 1, 2, \dots, r/3-2$)

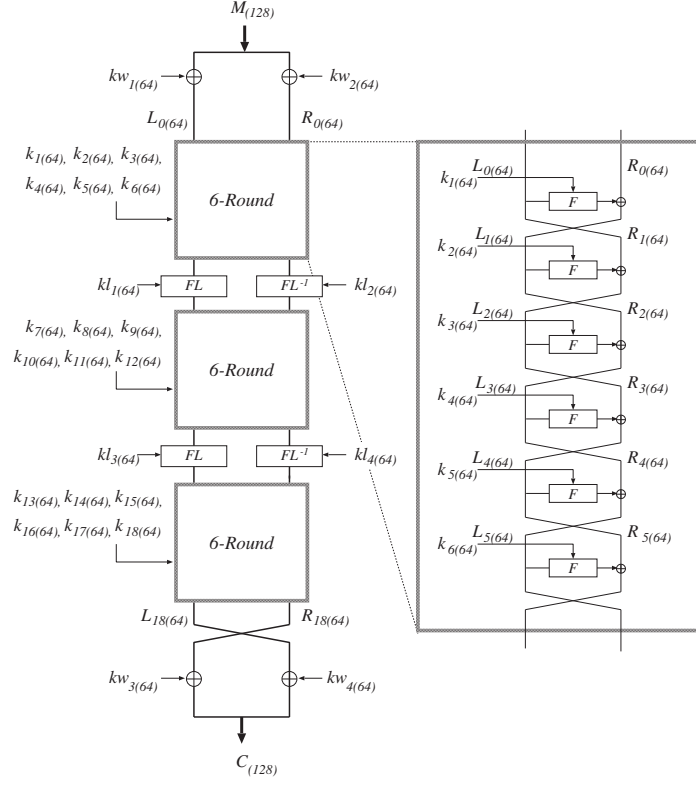


Fig. 1. Encryption procedure of Camellia for 128-bit keys

for FL - and FL^{-1} -functions from the secret key K , where r is the number of rounds.

2.1 Notations

X_L , X_R : the left-half and the right-half data of X , respectively.

\oplus, \cap, \cup : bitwise exclusive-OR (XOR), AND and OR operation, respectively.

\parallel : concatenation of two operands.

\gg_n, \ll_n : rotation to the right and the left by n bits, respectively.

$0x$: hexadecimal representation.

2.2 Encryption for 128-Bit Keys

First a 128-bit plaintext M is XORed with $kw_1 \parallel kw_2$ and separated into two 64-bit data L_0 and R_0 , i.e., $M \oplus (kw_1 \parallel kw_2) = L_0 \parallel R_0$. Then, the following operations are performed from $r = 1$ to 18, except for $r = 6$ and 12;

$$L_r = R_{r-1} \oplus F(L_{r-1}, k_r), \quad R_r = L_{r-1}.$$

For $r = 6$ and 12 , the following is carried out;

$$\begin{aligned} L'_r &= R_{r-1} \oplus F(L_{r-1}, k_r), & R'_r &= L_{r-1}, \\ L_r &= FL(L'_r, kl_{r/3-1}), & R_r &= FL^{-1}(R'_r, kl_{r/3}). \end{aligned}$$

Lastly, R_{18} and L_{18} are concatenated and XORed with $kw_3||kw_4$. The resultant value is the 128-bit ciphertext, i.e., $C = (R_{18}||L_{18}) \oplus (kw_3||kw_4)$.

2.3 Encryption for 192- and 256-Bit Keys

Similarly to the encryption for 128-bit keys, first a 128-bit plaintext M is XORed with $kw_1||kw_2$ and separated into two 64-bit data L_0 and R_0 , i.e., $M \oplus (kw_1||kw_2) = L_0||R_0$. Then, the following operations are performed from $r = 1$ to 24 , except for $r = 6, 12$, and 18 ;

$$L_r = R_{r-1} \oplus F(L_{r-1}, k_r), \quad R_r = L_{r-1}.$$

For $r = 6, 12$, and 18 , the following are performed;

$$\begin{aligned} L'_r &= R_{r-1} \oplus F(L_{r-1}, k_r), & R'_r &= L_{r-1}, \\ L_r &= FL(L'_r, kl_{r/3-1}), & R_r &= FL^{-1}(R'_r, kl_{r/3}). \end{aligned}$$

Lastly, R_{24} and L_{24} are concatenated and XORed with $kw_3||kw_4$. The resultant value is the 128-bit ciphertext, i.e., $C = (R_{24}||L_{24}) \oplus (kw_3||kw_4)$.

2.4 Decryption

The decryption procedure of Camellia can be done in the same way as the encryption procedure by reversing the order of the subkeys, which is one of merits of Feistel networks. In Camellia, FL/FL^{-1} -function layers are inserted every 6 rounds, but this property is still preserved.

2.5 Key Schedule

Figure 2 shows the key schedule of Camellia. Two 128-bit variables K_L and K_R are defined as follows. For 128-bit keys, the 128-bit key K is used as K_L and K_R is 0. For 192-bit keys, the left 128-bit of the key K is used as K_L , and concatenation of the right 64-bit of K and the complement of the right 64-bit of K is used as K_R . For 256-bit keys, the left 128-bit of the key K is used as K_L and the right 128-bit of K is used as K_R .

Two 128-bit variables K_A and K_B are generated from K_L and K_R as shown in Fig. 2. Note that K_B is used only if the length of the secret key is 192 or 256 bits. The 64-bit constants Σ_i ($i = 1, 2, \dots, 6$) are used as “keys” in the Feistel network. They are defined as continuous values from the second hexadecimal place to the seventeenth hexadecimal place of the hexadecimal representation of the square root of the i -th prime. These constant values are shown in Table 1.

The 64-bit subkeys kw_t , kw_u , and kl_v are generated from K_L , K_R , K_A , and K_B . The subkeys are generated by rotating K_L , K_R , K_A , and K_B and taking the left- or right-half of them. Details are shown in Table 2.

Table 1. The key schedule constants

| | | | |
|------------|--------------------|------------|--------------------|
| Σ_1 | 0xA09E667F3BCC908B | Σ_4 | 0x54FF53A5F1D36F1C |
| Σ_2 | 0xB67AE8584CAA73B2 | Σ_5 | 0x10E527FADE682D1D |
| Σ_3 | 0xC6EF372FE94F82BE | Σ_6 | 0xB05688C2B3E6C1FD |

Table 2. Subkeys for 128-bit keys and 192/256-bit keys

| 128-bit keys | subkey | value | 192/256-bit keys | subkey | value |
|----------------|----------|--------------------|------------------|----------|--------------------|
| Prew whitening | kw_1 | $(K_L \lll 0)_L$ | Prew whitening | kw_1 | $(K_L \lll 0)_L$ |
| | kw_2 | $(K_L \lll 0)_R$ | | kw_2 | $(K_L \lll 0)_R$ |
| F (Round1) | k_1 | $(K_A \lll 0)_L$ | F (Round1) | k_1 | $(K_B \lll 0)_L$ |
| F (Round2) | k_2 | $(K_A \lll 0)_R$ | F (Round2) | k_2 | $(K_B \lll 0)_R$ |
| F (Round3) | k_3 | $(K_L \lll 15)_L$ | F (Round3) | k_3 | $(K_R \lll 15)_L$ |
| F (Round4) | k_4 | $(K_L \lll 15)_R$ | F (Round4) | k_4 | $(K_R \lll 15)_R$ |
| F (Round5) | k_5 | $(K_A \lll 15)_L$ | F (Round5) | k_5 | $(K_A \lll 15)_L$ |
| F (Round6) | k_6 | $(K_A \lll 15)_R$ | F (Round6) | k_6 | $(K_A \lll 15)_R$ |
| FL | kl_1 | $(K_A \lll 30)_L$ | FL | kl_1 | $(K_R \lll 30)_L$ |
| FL^{-1} | kl_2 | $(K_A \lll 30)_R$ | FL^{-1} | kl_2 | $(K_R \lll 30)_R$ |
| F (Round7) | k_7 | $(K_L \lll 45)_L$ | F (Round7) | k_7 | $(K_B \lll 30)_L$ |
| F (Round8) | k_8 | $(K_L \lll 45)_R$ | F (Round8) | k_8 | $(K_B \lll 30)_R$ |
| F (Round9) | k_9 | $(K_A \lll 45)_L$ | F (Round9) | k_9 | $(K_L \lll 45)_L$ |
| F (Round10) | k_{10} | $(K_L \lll 60)_R$ | F (Round10) | k_{10} | $(K_L \lll 45)_R$ |
| F (Round11) | k_{11} | $(K_A \lll 60)_L$ | F (Round11) | k_{11} | $(K_A \lll 45)_L$ |
| F (Round12) | k_{12} | $(K_A \lll 60)_R$ | F (Round12) | k_{12} | $(K_A \lll 45)_R$ |
| FL | kl_3 | $(K_L \lll 77)_L$ | FL | kl_3 | $(K_L \lll 60)_L$ |
| FL^{-1} | kl_4 | $(K_L \lll 77)_R$ | FL^{-1} | kl_4 | $(K_L \lll 60)_R$ |
| F (Round13) | k_{13} | $(K_L \lll 94)_L$ | F (Round13) | k_{13} | $(K_R \lll 60)_L$ |
| F (Round14) | k_{14} | $(K_L \lll 94)_R$ | F (Round14) | k_{14} | $(K_R \lll 60)_R$ |
| F (Round15) | k_{15} | $(K_A \lll 94)_L$ | F (Round15) | k_{15} | $(K_B \lll 60)_L$ |
| F (Round16) | k_{16} | $(K_A \lll 94)_R$ | F (Round16) | k_{16} | $(K_B \lll 60)_R$ |
| F (Round17) | k_{17} | $(K_L \lll 111)_L$ | F (Round17) | k_{17} | $(K_L \lll 77)_L$ |
| F (Round18) | k_{18} | $(K_L \lll 111)_R$ | F (Round18) | k_{18} | $(K_L \lll 77)_R$ |
| Post whitening | kw_3 | $(K_A \lll 111)_L$ | FL | kl_5 | $(K_A \lll 77)_L$ |
| | kw_4 | $(K_A \lll 111)_R$ | FL^{-1} | kl_6 | $(K_A \lll 77)_R$ |
| | | | F (Round19) | k_{19} | $(K_R \lll 94)_L$ |
| | | | F (Round20) | k_{20} | $(K_R \lll 94)_R$ |
| | | | F (Round21) | k_{21} | $(K_A \lll 94)_L$ |
| | | | F (Round22) | k_{22} | $(K_A \lll 94)_R$ |
| | | | F (Round23) | k_{23} | $(K_L \lll 111)_L$ |
| | | | F (Round24) | k_{24} | $(K_L \lll 111)_R$ |
| | | | Post whitening | kw_3 | $(K_B \lll 111)_L$ |
| | | | | kw_4 | $(K_B \lll 111)_R$ |

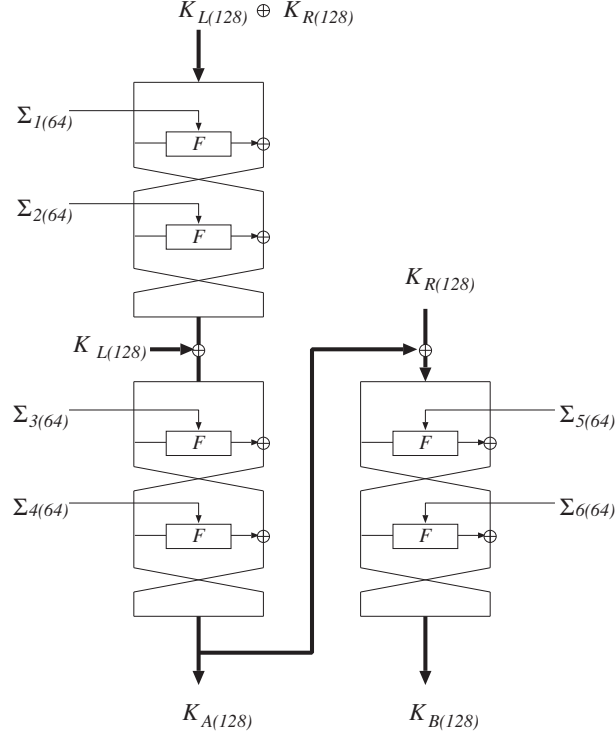


Fig. 2. Key schedule

3 Components of Camellia

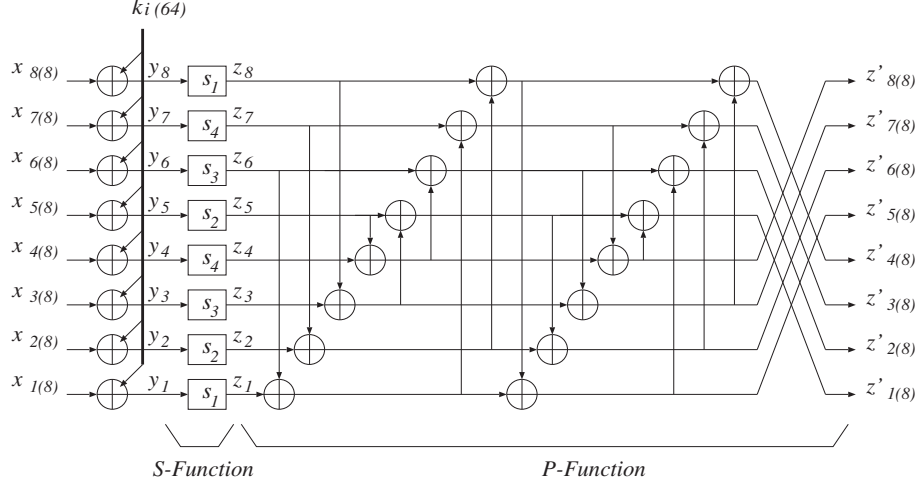
3.1 F -Function

The F -function is shown in Fig. 3. The F -function uses the SPN (Substitution-Permutation Network) structure. The S -function is the non-linear layer and the P -function is the linear layer.

3.2 S -Function, s -Boxes

The S -function consists of eight s -boxes, and four different s -boxes, s_1 , s_2 , s_3 , and s_4 are used. All of them are affine equivalent to the inversion function in $\text{GF}(2^8)$. The data of s_2 , s_3 , and s_4 can be generated from the s_1 table. The tables are shown in [1].

$$\begin{aligned}
 s_1 &: \text{GF}(2)^8 \rightarrow \text{GF}(2)^8, x \mapsto \mathbf{h}(\mathbf{g}(\mathbf{f}(0\mathbf{x}c5 \oplus x))) \oplus 0\mathbf{x}6\mathbf{e} \\
 s_2 &: \text{GF}(2)^8 \rightarrow \text{GF}(2)^8, x \mapsto s_1(x) \lll_1 \\
 s_3 &: \text{GF}(2)^8 \rightarrow \text{GF}(2)^8, x \mapsto s_1(x) \ggg_1 \\
 s_4 &: \text{GF}(2)^8 \rightarrow \text{GF}(2)^8, x \mapsto s_1(x \lll_1)
 \end{aligned}$$

Fig. 3. *F*-function

where functions **f** and **h** are affine functions and function **g** is the inversion function in $\text{GF}(2^8)$ as given below.

$$\mathbf{f} : \text{GF}(2)^8 \rightarrow \text{GF}(2)^8, (a_1, a_2, \dots, a_8) \mapsto (b_1, b_2, \dots, b_8),$$

where

$$\begin{aligned} b_1 &= a_6 \oplus a_2, & b_2 &= a_7 \oplus a_1, & b_3 &= a_8 \oplus a_5 \oplus a_3, & b_4 &= a_8 \oplus a_3, \\ b_5 &= a_7 \oplus a_4, & b_6 &= a_5 \oplus a_2, & b_7 &= a_8 \oplus a_1, & b_8 &= a_6 \oplus a_4. \end{aligned}$$

$$\mathbf{h} : \text{GF}(2)^8 \rightarrow \text{GF}(2)^8, (a_1, a_2, \dots, a_8) \mapsto (b_1, b_2, \dots, b_8),$$

where

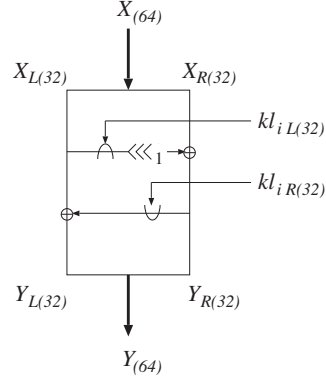
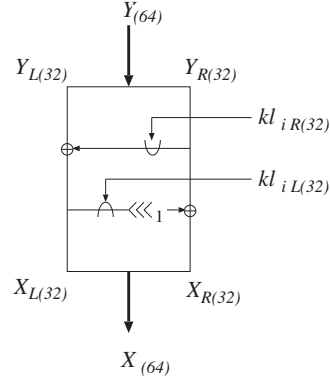
$$\begin{aligned} b_1 &= a_5 \oplus a_6 \oplus a_2, & b_2 &= a_6 \oplus a_2, & b_3 &= a_7 \oplus a_4, & b_4 &= a_8 \oplus a_2, \\ b_5 &= a_7 \oplus a_3, & b_6 &= a_8 \oplus a_1, & b_7 &= a_5 \oplus a_1, & b_8 &= a_6 \oplus a_3. \end{aligned}$$

$$\mathbf{g} : \text{GF}(2)^8 \rightarrow \text{GF}(2)^8, (a_1, a_2, \dots, a_8) \mapsto (b_1, b_2, \dots, b_8),$$

where

$$\begin{aligned} & (b_8 + b_7\alpha + b_6\alpha^2 + b_5\alpha^3) + (b_4 + b_3\alpha + b_2\alpha^2 + b_1\alpha^3)\beta \\ &= ((a_8 + a_7\alpha + a_6\alpha^2 + a_5\alpha^3) + (a_4 + a_3\alpha + a_2\alpha^2 + a_1\alpha^3)\beta)^{-1}. \end{aligned}$$

This inversion is performed in $\text{GF}(2^8)$ assuming $0^{-1} = 0$, where β is an element in $\text{GF}(2^8)$ that satisfies $\beta^8 + \beta^6 + \beta^5 + \beta^3 + 1 = 0$, and $\alpha = \beta^{238} = \beta^6 + \beta^5 + \beta^3 + \beta^2$ is an element in $\text{GF}(2^4)$ that satisfies $\alpha^4 + \alpha + 1 = 0$.

Fig. 4. FL -functionFig. 5. FL^{-1} -function

3.3 P -Function

The P -function is defined as follows:

$$P : (\text{GF}(2)^8)^8 \rightarrow (\text{GF}(2)^8)^8, (z_1, z_2, \dots, z_8) \mapsto (z'_1, z'_2, \dots, z'_8),$$

where

$$\begin{aligned} z'_1 &= z_1 \oplus z_3 \oplus z_4 \oplus z_6 \oplus z_7 \oplus z_8, & z'_2 &= z_1 \oplus z_2 \oplus z_4 \oplus z_5 \oplus z_7 \oplus z_8, \\ z'_3 &= z_1 \oplus z_2 \oplus z_3 \oplus z_5 \oplus z_6 \oplus z_8, & z'_4 &= z_2 \oplus z_3 \oplus z_4 \oplus z_5 \oplus z_6 \oplus z_7, \\ z'_5 &= z_1 \oplus z_2 \oplus z_6 \oplus z_7 \oplus z_8, & z'_6 &= z_2 \oplus z_3 \oplus z_5 \oplus z_7 \oplus z_8, \\ z'_7 &= z_3 \oplus z_4 \oplus z_5 \oplus z_6 \oplus z_8, & z'_8 &= z_1 \oplus z_4 \oplus z_5 \oplus z_6 \oplus z_7. \end{aligned}$$

3.4 FL -Function and FL^{-1} -Function

The FL -function is shown in Fig. 4, and is defined as follows.

$$FL : \text{GF}(2)^{64} \times \text{GF}(2)^{64} \rightarrow \text{GF}(2)^{64}, (X_L || X_R, kl_L || kl_R) \mapsto Y_L || Y_R,$$

where

$$Y_R = ((X_L \cap kl_L) \lll_1) \oplus X_R, \quad Y_L = (Y_R \cup kl_R) \oplus X_L.$$

The FL^{-1} -function is shown in Fig. 5. The following equation holds.

$$FL^{-1}(FL(x, k), k) = x.$$

4 Design Rationale

4.1 F -Function

The design strategy of the F -function of Camellia follows that of the F -function of E2 [14]. The main difference between E2 and Camellia is the adoption of

the 1-round (conservative) SPN, not the 2-round SPN, i.e., S-P-S. When the 1-round SPN is used as the round function in a Feistel cipher, the theoretical evaluation of the upper bound of differential and linear characteristic probability becomes more complicated, but the speed under the same level of “real” security is expected to be improved. See Sect. 6 for detailed discussions on security.

4.2 *P*-Function

The design rationale of the *P*-function is similar to that of the *P*-function of E2 [16]. That is, for computational efficiency, it should be represented using only bitwise XORs and for security against differential and linear cryptanalyses, its branch number should be optimal. From among the linear transformations that satisfy these conditions, we chose one considering highly efficient implementation on 32-processors [3] and high-end smart cards, as well as 8-bit processors.

4.3 *s*-Boxes

As the *s*-boxes we adopted functions affine equivalent to the inversion function in $\text{GF}(2^8)$ for enhanced security and small hardware design.

There is a function affine equivalent to the inversion function in $\text{GF}(2^8)$ that achieves the best known of the maximum differential and linear probabilities, 2^{-6} . We choose this kind of functions as *s*-boxes. Moreover, the high degree of the Boolean polynomial of every output bit of the *s*-boxes makes it difficult to attack Camellia by higher order differential attacks. The two affine functions that are performed at the input and output of the inversion function in $\text{GF}(2^8)$ complicates the expressions of the *s*-boxes in $\text{GF}(2^8)$, which is expected to make interpolation attacks ineffective. Making the four *s*-boxes different slightly improves security against truncated differential cryptanalysis [23].

For small hardware design, the elements in $\text{GF}(2^8)$ can be represented as polynomials with coefficients in the subfield $\text{GF}(2^4)$. In other words, we can implement the *s*-boxes by using a few operations in the subfield $\text{GF}(2^4)$ [22]. Two affine functions at the input and output of the inversion function in $\text{GF}(2^8)$ also play a role in complicating the expressions of the *s*-boxes in $\text{GF}(2^4)$.

4.4 *FL*- and *FL*⁻¹-Functions

FL- and *FL*⁻¹-functions are “inserted” between every 6 rounds of a Feistel network to provide non-regularity across rounds. One of the goals for such a design is to thwart future unknown attacks. It is one of merits of regular Feistel networks that encryption and decryption procedures are the same except for the order of the subkeys. In Camellia, *FL/FL*⁻¹-function layers are inserted every 6 rounds, but this property is still preserved.

The design criteria of these functions are similar to those of the *FL*-function of MISTY [20]. The difference between MISTY and Camellia is the addition of 1-bit rotation. This is expected to make bitwise cryptanalysis harder, but it

has no negative impact on hardware size or speed. The design criteria are that these functions must be linear for any fixed key and that their forms depend on key values. Since these functions are linear as long as the key is fixed, they do not make the average differential and linear probabilities of the cipher higher. Moreover, these functions are fast in both software and hardware since they are constructed by logical operations such as AND, OR, XOR, and rotations.

4.5 Key Schedule

The design criteria of the key schedule are as follows.

1. It should be simple and share part of its procedure with encryption (and decryption).
2. Subkey generation for 128-, 192- and 256-bit keys can be performed by using the same key schedule (circuit). Moreover, the key schedule for 128-bit keys can be performed by using a part of this circuit.
3. Key setup time should be shorter than encryption time. In cases where large amounts of data are processed with a single secret key, the setup time for key scheduling may be unimportant. On the other hand, in applications in which the key is changed frequently, key agility is a factor. One basic component of key agility is key setup time.
4. It should support on-the-fly subkey generation.
5. On-the-fly subkey generation should be computable in the same way in both encryption and decryption. Some ciphers have separate key schedules for encryption and decryption. In other ciphers, e.g., Rijndael or Serpent, subkeys are computable in the forward direction only and require unwinding for decryption.
6. There should be no equivalent keys.
7. There should be no related-key attacks or slide attacks.

Criteria 1 and 2 mainly address small hardware requirements, Criteria 3, 4, and 5 are advantageous in terms of practical applications, and Criteria 6 and 7 are for security.

The memory requirement for generating subkeys is quite small. An efficient implementation of Camellia for 128-bit keys requires 16 bytes (=128 bits) for the original secret key, K_L , and 16 bytes (=128 bits) for the intermediate key, K_A . Thus the required memory is 32 bytes. Similarly, an efficient implementation of Camellia for 192- and 256-bit keys needs only 64 bytes.

5 Security

This section discusses the security of Camellia. Hereafter, we call Camellia without FL - and FL^{-1} -functions Camellia*.

5.1 Differential and Linear Cryptanalysis

The most well-known and powerful approaches to attacking many block ciphers are differential cryptanalysis [4] and linear cryptanalysis [18]. There are several methods of evaluating security against these attacks, where there is a kind of “duality” relation between them [19,8]: in other words, the security against both attacks can be evaluated in similar ways.

It is known that the upper bounds of differential and linear characteristic probabilities can, for several block ciphers, be estimated using the minimum numbers of differential and linear active s -boxes in some consecutive rounds, respectively. Kanda [11] shows the minimum numbers of differential and linear active s -boxes for Feistel ciphers with conservative SPN (S-P) round function.

Definition 1. ([25]) *The branch number \mathcal{B} of linear transformation P is defined by*

$$\mathcal{B} = \min_{x \neq 0} (w_H(x) + w_H(P(x))),$$

where $w_H(x)$ denotes the bitwise Hamming weight of x .

Theorem 1. *The minimum number of differential/linear active s -boxes in any eight consecutive rounds is equal or larger than $2\mathcal{B} + 1$.*

Theorem 2. *Let p_s and q_s be the maximum differential and linear probabilities of all s -boxes, and \mathcal{D} and \mathcal{L} be the minimum numbers of total differential and linear active s -boxes, respectively. Then, the maximum differential and linear characteristic probabilities are bounded by $p_s^{\mathcal{D}}$ and $q_s^{\mathcal{L}}$, respectively.*

In the case of Camellia, the maximum differential and linear probabilities of the s -boxes are $p_s = q_s = 2^{-6}$. The branch number of the linear transformation (P -function) is 5, i.e., $\mathcal{B} = 5$. Letting p, q be the maximum differential and linear characteristic probabilities of Camellia* reduced to 16-round, respectively, we have $p \leq p_s^{2(2\mathcal{B}+1)} = (2^{-6})^{22} = 2^{-132}$ and $q \leq q_s^{2(2\mathcal{B}+1)} = (2^{-6})^{22} = 2^{-132}$ from Theorems 1 and 2. Both probabilities are below the security threshold of 128-bit block ciphers: 2^{-128} . It follows that there is no effective differential characteristic or linear characteristic for Camellia* reduced to more than 15 rounds. Since FL - and FL^{-1} -functions are linear for any fixed key, they do not make the average differential and linear probabilities of the cipher higher. Hence, it is proven that Camellia offers enough security against differential and linear cryptanalyses.

Note that the result above are based on Theorems 1 and 2. Both theorems deal with general cases of Feistel ciphers with SPN round function, so we expect that Camellia is actually more secure than shown by the result above. As supporting evidence, we counted the number of active s -boxes of Camellia and Camellia* with reduced rounds. The counting algorithm is similar to that described in [21] except following three items.

- Prepare the table for the number of active s -boxes instead of transition probability table.

Table 3. Upper bounds of differential characteristic probability of Camellia

| # of rounds | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 |
|-------------|-----|----------|-----------|-----------|-----------|-----------|-----------|-----------|------------|------------|------------|------------|
| Based on | | | 2^{-12} | 2^{-30} | | 2^{-42} | | 2^{-66} | | | | 2^{-96} |
| Th. 1 and 2 | | | (2) | (5) | | (7) | | (11) | | | | (16) |
| Camellia | 1 | 2^{-6} | 2^{-12} | 2^{-42} | 2^{-54} | 2^{-66} | 2^{-72} | 2^{-72} | 2^{-78} | 2^{-108} | 2^{-120} | 2^{-132} |
| | (0) | (1) | (2) | (7) | (9) | (11) | (12) | (12) | (13) | (18) | (20) | (22) |
| Camellia* | 1 | 2^{-6} | 2^{-12} | 2^{-36} | 2^{-54} | 2^{-66} | 2^{-78} | 2^{-90} | 2^{-108} | 2^{-126} | 2^{-132} | |
| | (0) | (1) | (2) | (6) | (9) | (11) | (13) | (15) | (18) | (21) | (22) | |

Note: The numbers in brackets are the number of active s-boxes.

Table 4. Upper bounds of linear characteristic probability of Camellia

| # of rounds | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 |
|-------------|-----|----------|-----------|-----------|-----------|-----------|-----------|-----------|------------|------------|------------|------------|
| Based on | | | 2^{-12} | 2^{-30} | | 2^{-42} | | 2^{-66} | | | | 2^{-96} |
| Th. 1 and 2 | | | (2) | (5) | | (7) | | (11) | | | | (16) |
| Camellia | 1 | 2^{-6} | 2^{-12} | 2^{-36} | 2^{-54} | 2^{-66} | 2^{-72} | 2^{-72} | 2^{-78} | 2^{-102} | 2^{-120} | 2^{-132} |
| | (0) | (1) | (2) | (6) | (9) | (11) | (12) | (12) | (13) | (17) | (20) | (22) |
| Camellia* | 1 | 2^{-6} | 2^{-12} | 2^{-36} | 2^{-54} | 2^{-66} | 2^{-78} | 2^{-84} | 2^{-108} | 2^{-120} | 2^{-132} | |
| | (0) | (1) | (2) | (6) | (9) | (11) | (13) | (14) | (18) | (20) | (22) | |

Note: The numbers in brackets are the number of active s-boxes.

- Count the number of active s -boxes instead of computing transition probability.
- FL - and FL^{-1} -functions set all elements to the minimum number of active s -boxes in the table. This means that the algorithm gives consideration to existence of weak subkeys inserted to FL - and FL^{-1} -functions, since there may be some possibility of connecting every later differential and linear characteristic with the previous one with the highest probability, which is equivalent to the minimum number of active s -boxes.

As a result, we confirmed that 12-round Camellia has no differential and linear characteristic with probability higher than 2^{-128} (see Tables 3 and 4).

5.2 Truncated Differential and Linear Cryptanalysis

The attacks using truncated differentials were introduced by Knudsen [13]. He defined them as differentials where only a part of the difference can be predicted. The notion of truncated differentials introduced by him is wide, but with a byte-oriented cipher it is natural to study bitwise differentials as truncated differentials [23].

The maximum differential probability is considered to provide the strict evaluation of security against differential cryptanalysis, but computing its value is impossible in general, since a differential is a set of all differential characteristics with the same input difference and the same output difference for a Markov cipher [17]. On the other hand, a truncated differential can be regarded as a subset

of the differential characteristics which are exploitable in cryptanalysis. For some ciphers, e.g., byte-oriented ciphers, the probability of truncated differential can be computed easily and correctly, and it gives a more strict evaluation than the maximum differential characteristic probability.

A truncated differential cryptanalysis of reduced-round variants of E2 was presented by Matsui and Tokita at FSE'99 [23]. Their analysis was based on the “byte characteristic,” where the values to the difference in a byte are distinguished between non-zero and zero. They found a 7-round byte characteristic, which leads to a possible attack on an 8-round variant of E2 without *IT*-Function (the initial transformation) and *FT*-Function (the final transformation). The best attack of E2 shown in [24] breaks an 8-round variant of E2 with either *IT*-Function or *FT*-Function using 2^{94} chosen plaintexts. In [24] we also show the attack which distinguishes a 7-round variant of E2 with *IT*- and *FT*-Functions from a random permutation using 2^{91} chosen plaintexts.

Camellia is a byte-oriented cipher similar to E2, and it is important to evaluate its security against truncated differential cryptanalysis. We searched for truncated differentials using an algorithm similar to the one described in [23,24]. The main difference of the round function between E2 and Camellia is the adoption of the 1-round SPN not the 2-round SPN, i.e., S-P-S. In the search for truncated differentials of E2, we used about 2^{-8} as the probability of difference cancellation in one byte at the XOR of Feistel network. However, the round function of Camellia doesn't have the second *s*-boxes-layer, and the difference cancellation in plural bytes sometimes occurs with the same probability. Accordingly, we changed the difference cancellation rule at the XOR of Feistel network in the search algorithm. As a result, Camellia with more than 10 rounds is indistinguishable from a random permutation, in both cases with/without *FL*/*FL*⁻¹-function layers.

Next, we introduce a new cryptanalysis called truncated linear cryptanalysis. Due to the duality between differential and linear cryptanalyses, we can evaluate security against truncated linear cryptanalysis by using a similar algorithm to that above. To put it concretely, we can perform the search by replacing the matrix of *P*-function with the transposed matrix. As a result, Camellia* with more than 10 rounds is indistinguishable from a random permutation.

5.3 Boomerang Attack

Boomerang attack [26] requires two differentials. Let the probability of the differentials be p_{Δ} and p_{∇} . An boomerang attack that is superior than exhaustive key search requires

$$p_{\Delta}p_{\nabla} \geq 2^{-64}. \quad (1)$$

Using Table 3, there is no combination that satisfies (1) for Camellia*. The best boomerang probability for Camellia* reduced to 8-round is bounded by 2^{-66} that is obtained by $p_{\Delta} = 2^{-12}$ (3 rounds) and $p_{\nabla} = 2^{-54}$ (5 rounds). Since the attackable rounds is bounded by much shorter than the specification of Camellia, 18 or 24, Camellia seems secure against a boomerang attack.

Table 5. Smallest number of unknown coefficients for 128-, 192-, and 256-bit keys

| | |
|---|-----|
| whitening $\times 1$ + round $\times r$ ($r < 4$) | 1 |
| whitening $\times 1$ + round $\times 4$ | 255 |
| More rounds | 256 |

5.4 Higher Order Differential Attack

Higher order differential attack is generally applicable to ciphers that can be represented as Boolean polynomials of low degree. All intermediate bits in the encryption process can be represented as Boolean polynomials, i.e., polynomials $\text{GF}(2)[x_1, x_2, \dots, x_n]$ in the bits of the plaintext: $\{x_1, x_2, \dots, x_n\}$. In the higher order differential attack described in [10, Theorem 1], if the intermediate bits are represented by Boolean polynomials of degree at least d , the $(d+1)$ -th order differential of the Boolean polynomial becomes 0.

For the degrees of Boolean polynomials of the s -boxes of Camellia, the functions affine equivalent to the inversion function in $\text{GF}(2^8)$ are adopted as the s -boxes. We confirmed that the degree of the Boolean polynomial of every output bit of the s -boxes is 7 by finding Boolean polynomial for every output bit of the s -boxes. In Camellia, it is expected that the degree of an intermediate bit in the encryption process increases as the data pass through many s -boxes. For example, the degree becomes $7^3 > 128$ after passing through three s -boxes. Therefore, we expect that higher order differential attacks fail against Camellia with full rounds.

5.5 Interpolation Attack and Linear Sum Attack

The interpolation attack proposed in [10] is typically applicable to attacking ciphers that use simple algebraic functions. Linear sum attack [2] is a generalization of the interpolation attack.

A practical algorithm that evaluates the security against linear sum attack was proposed in [2]. We searched for linear relations between any plaintext byte and any ciphertext byte over $\text{GF}(2^8)$ using the algorithm. Table 5 summarizes the results, and shows that Camellia is secure against linear sum attack including interpolation attack. It also implies that Camellia is secure against SQUARE attack [9] followed by [2, Theorem 3].

5.6 Security of Key Schedule

No Equivalent Keys: Since the set of subkeys generated by the key schedule contain the original secret key, there is no equivalent set of subkeys generated from distinct secret keys. Therefore, we expect that there are no distinct secret keys both of which encrypt each of many plaintexts into the same ciphertext.

Slide Attack: In [6,7] the slide attacks were introduced, based on earlier work in [5,12]. In particular it was shown that iterated ciphers with identical round

functions, that is, equal structures and equal subkeys in the round functions, are susceptible to slide attacks.

In Camellia, FL - and FL^{-1} -functions are “inserted” between every 6 rounds of a Feistel network to provide non-regularity across rounds. Moreover, from the viewpoint of the key schedule, slide attacks seems to be very unlikely to succeed.

Related-Key Attack: We are convinced that the key schedule of Camellia makes related-key attacks [5,15] very difficult. In these attacks, an attacker must be able to get encryptions using several related keys. However, since the subkeys depend on K_A and K_B , which are the results of encryption of a secret key, and if an attacker wants to change the secret key, he can’t get K_A and K_B desired, and vice versa, these subkey relations will be very hard to control and predict.

6 Performance

6.1 Software Implementations

Table 6 summarizes the current software implementations of Camellia. The table shows that Camellia can be efficiently implemented on low-end smart cards, and 32-bit and 64-bit processors. We use the abbreviations M (mega) for 10^6 and m (milli) for 10^{-3} in the table.

6.2 Hardware Performance

We measured the hardware performance of Camellia for 128-bit keys on ASIC (Application Specific Integrated Circuit) and FPGA (Field Programmable Gate Array). Table 7 shows the environment of our hardware design and evaluation. We evaluated hardware performance of the three types: Type 1, Type 2 and Type 3 logic. The hardware design policy of each type is as follows.

Type 1 Fast implementation from the viewpoint of encryption speed

Type 2 Small implementation from the viewpoint of total logic size

Type 3 Small implementation (special case for FPGA)

Tables 8 through 11 summarize the hardware performance of Camellia for 128-bit keys on ASIC and FPGA.

7 Conclusion

We have presented Camellia, the rationale behind its design, its suitability for both software and hardware implementation, and the results of our cryptanalyses. For further information, please refer to the specification of Camellia [1] or full paper, which are available on the Camellia home page: <http://info.isl.ntt.co.jp/camellia/>.

The performances shown in this paper leave room for further optimizations. The latest performance results will be posted on the Camellia home page.

Table 6. Camellia software performance

| Processor | Lang. | Key [bits] | Timing [cycles] | | Dynamic [bytes] | | Code [bytes] | | Table [bytes] |
|--------------------|----------------|---------------|-------------------------------------|------------------------------------|--------------------|-------------------|--------------------|-------------------|------------------|
| | | | Setup ^a (^b) | Enc. ^c (^d) | Setup ^a | Enc. ^c | Setup ^a | Enc. ^c | |
| P III ^e | Asm | 128 | 160 (4.4M) | 371 (242M) | 28 | 36 | 1,046 | 2,150 | 8,224 |
| | | 192 | 222 (3.2M) | 494 (181M) | 28 | 36 | 1,469 | 3,323 | 8,240 |
| | | 256 | 226 (3.1M) | 494 (181M) | 28 | 36 | 1,485 | 3,323 | 8,240 |
| P II ^f | C ^g | 128 | 263 (1.1M) | 577 (67M) | 44 | 64 | 1,600 | 3,733 | 4,128 |
| Alpha ^h | Asm | 128 | 118 (5.7M) | 339 (252M) | 48 | 48 | 1,132 | 3,076 | 16,528 |
| | | 192 | 176 (3.7M) | 445 (192M) | 48 | 48 | 1,668 | 4,000 | 16,528 |
| | | 256 | 176 (3.7M) | 445 (192M) | 48 | 48 | 1,676 | 4,000 | 16,528 |
| | | 128 | 158 (4.2M) | 326 (262M) | 48 | 48 | 1,600 | 2,928 | 16,512 |
| 8051 ⁱ | Asm | 128 | 0 (0) | 10217 (10m) | 0 | 32 | 0 | 702 | 288 |

^a Key schedule may be included.^b Seconds for 8051, and keys/s for other processors.^c Numbers of this column is the same as decryption.^d Seconds for 8051, and b/s for other processors.^e Intel Pentium III (700MHz), 256KB on-die L2 cache, FreeBSD 4.0R, 128MB main memory.^f Intel Pentium II (300MHz), 512KB L2 cache, MS-Windows 95, 160MB main memory.^g ANSI C, Microsoft Visual C++ 6 with the optimization options /G6 /Zp16 /ML /Ox /Ob2.^h Alpha 21264 (667MHz), Compaq Tru64 UNIX 4.0F, 2GB main memory.ⁱ Intel 8051 (12MHz; 1 cycle = 12 oscillator periods) simulator on Unix.**Table 7.** Hardware evaluation environment (ASIC, FPGA)

| | |
|-----------------|---|
| Language | (ASIC, FPGA) Verilog-HDL |
| Simulator | (ASIC, FPGA) Verilog-XL |
| Design library | (ASIC) Mitsubishi Electric 0.35 μ CMOS ASIC library (FPGA) Xilinx XC4000XL series |
| Login synthesis | (ASIC) Design Compiler version 1998.08 (FPGA) Synplify version 5.3.1 and ALLIANCE version 2.1i |

Table 8. Hardware performance (Type 1: [ASIC(0.35 μ CMOS)])

| Algorithm name | Area [Gate] | | | Key setup time [ns] | Critical- path [ns] | Throughput [Mb/s] |
|-------------------|------------------------|-------------------------|--------------------------|------------------------|------------------------|----------------------|
| | Enc.&Dec. ^a | Key expan. ^b | Total logic ^c | | | |
| DES | 42,204 | 12,201 | 54,405 | — | 55.11 | 1161.31 |
| Triple-DES | 124,888 | 23,207 | 128,147 | — | 157.09 | 407.40 |
| MARS | 690,654 | 2,245,096 | 2,935,754 | 1740.99 | 567.49 | 225.55 |
| RC6 | 741,641 | 901,382 | 1,643,037 | 2112.26 | 627.57 | 203.96 |
| Rijndael | 518,508 | 93,708 | 612,834 | 57.39 | 65.64 | 1950.03 |
| Serpent | 298,533 | 205,096 | 503,770 | 114.07 | 137.40 | 931.58 |
| Twofish | 200,165 | 231,682 | 431,857 | 16.38 | 324.80 | 394.08 |
| Camellia | 216,911 | 55,907 | 272,819 | 24.36 | 109.35 | 1170.55 |

^a including output registers^b including subkey registers^c including buffers for fan-out adjustment

Table 9. Hardware performance (Type 2: [ASIC(0.35 μ CMOS)])

| Algorithm name | Area [Gate] | | Key setup time [ns] | Critical- path [ns] | Throughput [Mb/s] |
|-------------------|------------------------|--|------------------------|------------------------|----------------------|
| | Enc.&Dec. ^a | Key sched. ^b Total logic ^c | | | |
| Camellia | 6,367 | 4,979 11,350 | 110.2 | 27.67 | 220.28 |

^a including output registers and data selector^b including subkey registers and a part of key expansion logic^c including buffers for fan-out adjustment**Table 10.** Hardware performance (Type 2: [FPGA(XC4000XL series)])

| Algorithm | Total Area [CLBs] | Critical-path [ns] | Throughput [Mb/s] |
|-----------|-------------------|--------------------|-------------------|
| Camellia | 1,296 | 78.815 | 77.34 |

Table 11. Hardware performance (Type 3: [FPGA(XC4000XL series)])

| Algorithm | Total Area [CLBs] | Critical-path [ns] | Throughput [Mb/s] |
|-----------|-------------------|--------------------|-------------------|
| Camellia | 874 | 49.957 | 122.01 |

We have analyzed Camellia and found no important weakness. The cipher has a conservative design and any practical attacks against Camellia would require a major breakthrough in the area of cryptanalysis. We think that Camellia is a very strong cipher, which matches the security of the existing best block ciphers.

References

1. Aoki, Ichikawa, Kanda, Matsui, Moriai, Nakajima, Tokita, "Specification of Camellia — a 128-bit Block Cipher," <http://info.isl.ntt.co.jp/camellia/>, 2000.
2. Aoki, "Practical Evaluation of Security against Generalized Interpolation Attack," IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences, Vol.E83-A, No.1, pp.33–38, 2000.
3. Aoki, Ueda, "Optimized Software Implementations of E2," IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences, Vol.E83-A, No.1, pp.101–105, 2000.
4. Biham, Shamir, "Differential Cryptanalysis of the Data Encryption Standard," Springer-Verlag, 1993.
5. Biham, "New Types of Cryptanalytic Attacks Using Related Keys," Journal of Cryptology, Vol.7, No.4, pp.229–246, Springer-Verlag, 1994.
6. Biryukov, Wagner, "Slide Attacks," Fast Software Encryption, FSE'99, LNCS 1636, pp.245–259, 1999.
7. Biryukov, Wagner, "Advanced Slide Attacks," Advances in Cryptology — EUROCRYPT2000, LNCS 1807, pp.589–606, 2000.
8. Chabaud, Vaudenay, "Links Between Differential and Linear Cryptanalysis," Advances in Cryptology — EUROCRYPT'94, LNCS 950, pp.356–365, 1995.
9. Daemen, Knudsen, Rijmen, "The Block Cipher SQUARE," Fast Software Encryption, FSE'97, LNCS 1267, pp.54–68, 1997.

10. Jakobsen, Knudsen, "The Interpolation Attack on Block Ciphers," Fast Software Encryption, FSE'97, LNCS 1267, pp.28–40, 1997.
11. Kanda, "Practical Security Evaluation against Differential and Linear Cryptanalyses for Feistel Ciphers with SPN Round Function," Selected Areas in Cryptography, SAC2000, LNCS in this proceeding.
12. Knudsen, "Cryptanalysis of LOKI91," Advances in Cryptology — AUSCRYPT'92, LNCS 718, pp.196–208, 1993.
13. Knudsen, "Truncated and Higher Order Differentials," Fast Software Encryption — Second International Workshop, LNCS 1008, pp.196–211, 1995.
14. Kanda, Moriai, Aoki, Ueda, Takashima, Ohta, Matsumoto, "E2 — A New 128-Bit Block Cipher," IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences, Vol.E83-A, No.1, pp.48–59, 2000.
15. Kelsey, Schneier, Wagner, "Key-Schedule Cryptanalysis of IDEA, G-DES, GOST, SAFER, and Triple-DES," Advances in Cryptology — CRYPTO'96, LNCS 1109, pp.237–251, 1996.
16. Kanda, Takashima, Matsumoto, Aoki, Ohta, "A Strategy for Constructing Fast Round Functions with Practical Security against Differential and Linear Cryptanalysis," Selected Areas in Cryptography, SAC'98, LNCS 1556, pp.264–279, 1999.
17. Lai, Massey, Murphy, "Markov Ciphers and Differential Cryptanalysis," Advances in Cryptology — EUROCRYPT'91, LNCS 547, pp.17–38, 1991.
18. Matsui, "Linear Cryptanalysis Method for DES Cipher," Advances in Cryptology — EUROCRYPT'93, LNCS 765, pp.386–397, 1994.
19. Matsui, "On Correlation Between the Order of S-boxes and the Strength of DES," Advances in Cryptology — EUROCRYPT'94, LNCS 950, pp.366–375, 1995.
20. Matsui, "New Block Encryption Algorithm MISTY," Fast Software Encryption, FSE'97, LNCS 1267, pp.54–68, 1997.
21. Matsui, "Differential Path Search of the Block Cipher E2," Technical report of IEICE, ISEC99-19, pp.57–64, The Institute of Electronics, Information and Communication Engineers, 1999. (in Japanese)
22. Matsui, Inoue, Yamagishi, Yoshida, "A note on calculation circuits over $GF(2^{2n})$," Technical Report of IEICE, IT88-14, The Institute of Electronics, Information and Communication Engineers, 1988. (in Japanese)
23. Matsui, Tokita, "Cryptanalysis of a Reduced Version of the Block Cipher E2," Fast Software Encryption, FSE'99, LNCS 1636, pp.71–80, 1999.
24. Moriai, Sugita, Aoki, Kanda, "Security of E2 against Truncated Differential Cryptanalysis," Selected Areas in Cryptography, SAC'99, LNCS 1758, pp.106–117, 2000.
25. Rijmen, Daemen, Preneel, Bosselaers, Win, "The cipher SHARK," Fast Software Encryption — Third International Workshop, LNCS 1039, pp.99–111, 1996.
26. Wagner, "The Boomerang Attack," Fast Software Encryption, FSE'99, LNCS 1636, pp.156–170, 1999.

DFCv2

Louis Granboulan¹, Phong Q. Nguyen¹, Fabrice Noilhan², and
Serge Vaudenay³

¹ École Normale Supérieure, Département d'Informatique
{Louis.Granboulan,Phong.Nguyen}@ens.fr

² Université d'Orsay, Laboratoire de Recherche en Informatique
Fabrice.Noilhan@ens.fr

³ Swiss Federal Institute of Technology (EPFL)
Serge.Vaudenay@epfl.ch

Abstract. The development process of the Advanced Encryption Standard (AES) was launched in 1997 by the US government through NIST. The Decorrelated Fast Cipher (DFC) was the CNRS proposal for the AES, among 14 other candidates in 1998. It was based on the recent decorrelation theory, to obtain certain security proofs covering linear and differential cryptanalysis. DFC received numerous comments. In particular, Coppersmith discovered a weakness in the key schedule. We address this weakness by a slight modification on DFC. This paper presents the specifications and rationales of DFC version 2, and discusses issues raised during the AES process.

1 Introduction

A major goal in cryptography is to prove security statements on encryption schemes. To this respect, it is well-known that the status of secret-key cryptography is quite different from that of public-key cryptography. The decorrelation theory was introduced in 1998 (see [20] for the original reference) as an attempt towards filling this gap, by providing new ideas to build block ciphers, together with security proofs covering certain (however general) classes of attacks. Since the AES process was launched by NIST at about the same period, the French National Center for Scientific Research (CNRS) decided to start a project aimed at showing that decorrelation theory was a reasonable proposal for making secure and efficient block ciphers. The target platform was chosen to be 64-bit microprocessors, as such chips are likely to become standard during the lifetime of the AES. The CNRS project gave birth to the “Decorrelated Fast Cipher” (DFC) [6,7].

Decorrelation theory (see [20,21,22,23,24,25]) enables to prove formal results on the security of cryptographic primitives under certain hypotheses which we believe to be realistic. In particular, it enables to quantify the best advantage to distinguish two families of block ciphers, for a class of attacks with limited resources. For instance, one can consider any Turing machine restricted to a given number d of oracle calls to the block cipher. Most of the existing block

ciphers are provably secure for the $d = 1$ case. However, none addresses the $d = 2$ case, except DFC and other decorrelation theory-based ones. Interestingly, the $d = 2$ case already provides formal security against possible formalizations of differential and linear cryptanalysis. The Nyberg-Knudsen approach [16] was the only previously known way to achieve similar security statements (with MISTY [13,14] as a famous example.) The MISTY approach however does not provide much design flexibility, and the DFC approach seems to achieve stronger results as shown in Section 4. Besides, the Nyberg-Knudsen approach is indeed an ad hoc construction for providing security against differential and linear attacks but does not consider other general attacks with $d = 2$.

Implementing decorrelated block ciphers with order $d = 2$ by using known techniques (like the PEANUT construction [20]) requires the use of built-in multiplication which leads to non-trivial optimization tricks. DFC was submitted to the AES in order to show that such challenges could be overcome. DFC attracted many comments from the AES community, sometimes controversial. For instance, it was claimed that DFC was too slow, that its security paradigm brought nothing new, and that the security margin was too small. In addition, Coppersmith discovered a weakness in the key schedule by showing the existence of a fraction of 2^{-128} of weak keys (using a quite complex algorithm).

In this paper, we give the complete specifications of DFCv2. This new version addresses the key schedule problem and allows scalable modifications of the internal structure (so that the user can choose any “security margin”). We also try to respond to the issues raised on the original DFC.

2 Specifications of DFCv2

In this section, we give the complete specifications of DFCv2, and emphasize rationales in each subsection. A sample test vector for the nominal choices of the parameters is given in Appendix.

2.1 Notation

All quantities are bit strings or integers. When string lengths are divisible by four, quantities are denoted in hexadecimal. For instance, `d43x` denotes the bitstring `110101000011` and also represents the (decimal) integer 3395 in arithmetic operations. We use classical bitwise bitstring operations: OR, AND, NOT, XOR. We also use the following arithmetic operations over the integers: $+$, \times , mod . The result of an arithmetic operation is implicitly converted into a bitstring whose length will be clear from the context. Finally, we use the bitstring concatenation $|$ and the trunc_n function that extracts the n leftmost bits of a bitstring.

2.2 High Level Overview

DFCv2 is characterized by four parameters m , k , r and s chosen for security and efficiency reasons. In $\text{DFCv2}(m, k, r, s)$, m is the message block length, k is the

key length, r is the number of encryption rounds, and s is the number of rounds for the subkey generation. We require that $m \geq 32$, $0 \leq k \leq 2m$, $rs \leq 128$, m is a multiple of 4, and r is even. The nominal choice for DFCv2 is $m = 128$, $k \in \{128, 192, 256\}$, $r = 8$ and $s = 4$.

The encryption function DFC_K operates on m -bit message blocks by means of a secret key K of arbitrary length k up to $2m$ bits. The corresponding decryption function is DFC_K^{-1} and operates on m -bit message blocks.

The secret key K is first turned into an mr -bit “Expanded Key” EK through an “Expanding Function” EF, *i.e.* $\text{EK} = \text{EF}(K)$. As explained in Section 2.5, the EF function applies r s -round Feistel schemes (see Feistel [5]). The encryption process itself performs a similar r -round Feistel scheme. Each round uses the “Round Function” RF. This function maps a $\frac{m}{2}$ -bit string onto a $\frac{m}{2}$ -bit string by using one m -bit string parameter. It is defined in Section 2.3.

Given a bitstring σ of length multiple of m , say $m\rho$, we split it into ρ m -bit strings

$$\sigma = p_1 | p_2 | \dots | p_\rho.$$

From σ we define a permutation Enc_σ on the set of m -bit strings coming from an ρ -round Feistel scheme. For any m -bit string PT which is split into two $\frac{m}{2}$ -bit halves x_0 and x_1 so that $\text{PT} = x_0 | x_1$. We build a sequence $x_0, \dots, x_{\rho+1}$ by the equation

$$x_{i+1} = \text{RF}_{p_i}(x_i) \text{ XOR } x_{i-1} \quad (i = 1, \dots, \rho) \quad (1)$$

and we define $\text{Enc}_\sigma(m) = x_{\rho+1} | x_\rho$.

Given an m -bit plaintext block PT and the mr -bit expanded key EK, the DFCv2_K encryption function is obtained as

$$\text{DFCv2}_K = \text{Enc}_{\text{EK}} \quad (2)$$

(that is, an r -round Feistel Cipher).

The EF function uses an s -round version defined with Enc.

If we split EK into r m -bit strings

$$\text{EK} = \text{RK}_1 | \text{RK}_2 | \dots | \text{RK}_r \quad (3)$$

obviously, we have $\text{DFC}_K^{-1} = \text{Enc}_{\text{revEK}}$ where

$$\text{revEK} = \text{RK}_r | \text{RK}_{r-1} | \dots | \text{RK}_1. \quad (4)$$

2.3 The RF Function

The RF function (as for “Round Function”) is fed with one m -bit parameter, which we view as two $\frac{m}{2}$ -bit parameters: an “ a -parameter” and a “ b -parameter”. It processes a $\frac{m}{2}$ -bit input x and outputs a $\frac{m}{2}$ -bit string defined as follows:

$$\text{RF}_{a|b}(x) = \text{CP}(((a \times x + b) \bmod p) \bmod 2^{\frac{m}{2}}) \quad (5)$$

where CP is a permutation over the set of all $\frac{m}{2}$ -bit strings (which appears in Section 2.4) and p is the smallest prime integer greater than $2^{\frac{m}{2}}$. For instance, if $m = 128$, we use $p = 2^{64} + 13$. See the following table for other values.

| m | p |
|-----|---------------|
| 32 | $2^{16} + 1$ |
| 64 | $2^{32} + 15$ |
| 96 | $2^{48} + 21$ |
| 128 | $2^{64} + 13$ |

Following the PEANUT scheme paradigm (see [20]), the RF function implements a decorrelation module. It is basically made from a classical round function (with CP), and from the pairwise decorrelation module $x \mapsto (ax + b \bmod p) \bmod 2^{\frac{m}{2}}$ which was used in the PEANUT construction.

From this construction, Decorrelation Theory ensures that if we consider DFCv2(128, $k, 6, s$) and if we make the heuristic assumption that EK is random and uniformly distributed from the random choice of the secret key, then the best advantage for distinguishing this reduced and idealized version of DFCv2 from a truly random permutation when limited to two chosen plaintexts is less than 2^{-117} (see [24]). This property has several consequences on the formal security of DFCv2 as summarized in Section 4.

2.4 The CP Permutation

The CP permutation (as for “Confusion Permutation”) uses a look-up table RT (as for “Round Table”) which takes a 6-bit integer as input and provides a $\frac{m}{4}$ -bit string output. Its size is thus $2m$ bytes.

Let $y = y_l | y_r$ be the input of CP where y_l and y_r are two $\frac{m}{4}$ -bit strings. We define

$$\text{CP}(y) = ((y_r \text{ XOR } (\text{RT} \circ \text{trunc}_6)(y_l)) | (y_l \text{ XOR } \text{KC})) + \text{KD} \bmod 2^{\frac{m}{2}} \quad (6)$$

where KC is a $\frac{m}{4}$ -bit constant string, and KD is a $\frac{m}{2}$ -bit constant string. The permutation CP is depicted in Fig. 1.

The constants $\text{RT}(0), \dots, \text{RT}(63)$, KC and KD will be set in Section 2.6.

The purpose of CP is to implement a permutation over all $\frac{m}{2}$ -bit strings which breaks the algebraic structure of the decorrelation module. For this we use a mixture of XORs and additions in a way very similar to that of the RC5 block cipher [19].

The RT tables play an important role by introducing randomness. These tables are limited to $2m$ bytes in total (in order to fit to embedded hardware with low memory) but with a maximal input size.

2.5 Key Scheduling Algorithm

In order to generate a sequence $\text{RK}_1, \text{RK}_2, \dots, \text{RK}_r$ from a given key K represented as a bit string of length at most $2m$, we use the following algorithm. We first pad K with a constant pattern KS in order to make a $2m$ -bit “Padded Key” string by

$$\text{PK} = \text{trunc}_{2m}(K | \text{KS}). \quad (7)$$

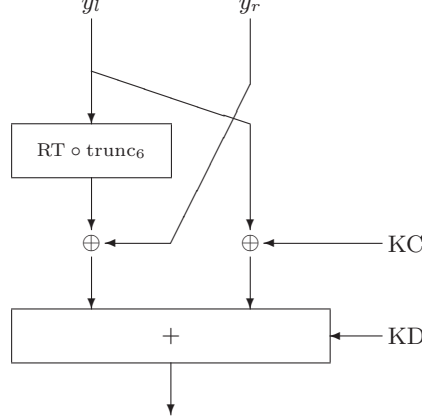


Fig. 1. The CP permutation

If K is of length m , we can observe that only the first m bits of KS are used. We define KS of length $2m$ in order to allow any key size from 0 to $2m$.

Then we split PK into two m -bit strings RK_0 and IRK_0 (as for “Internal Round Key”) such that $PK = IRK_0 | RK_0$ ¹. We assume we are given 16 m -bit constants KAB_0, \dots, KAB_{15} ². We now define

$$IRK_{j+1} = IRK_j \text{ XOR } \begin{cases} KAB_{RT(j) \bmod 16} & \text{if } j < 64 \\ KAB_{(RT(j-64) \gg 8) \bmod 16} & \text{otherwise} \end{cases} \quad (8)$$

for $j = 0, 1, \dots, rs - 1$ where $RT(j - 64) \gg 8$ denotes the bitstring $RT(j - 64)$ logically shifted by 8 bits to the right. Basically, we take the four least significant bits of $RT(j)$ for $j < 64$ and some other four bits of $RT(j - 64)$ for $64 \leq j < 128$. (Since we require that $rs \leq 128$, j is less than 128.) We notice that IRK_j is actually the XOR of IRK_0 with some constant depending on j .

Each sequence of s IRK_j values defines an sm -bit string IEK_i which serves as the round key sequence of some s -round internal encryption function. More precisely, we define

$$IEK_i = IRK_{is-s+1} | \dots | IRK_{is-1} | IRK_{is} \quad (9)$$

for $i = 1, 2, \dots, r$ and

$$IEnc_i = Enc_{IEK_i} \quad (10)$$

for $i = 1, 2, \dots, r$ as an “Internal Encryption”. We now define the RK_i sequence by

$$RK_i = IEnc_i(RK_{i-1}) \quad (11)$$

¹ The following IRK_i sequence replaces the $OAP_i | OBP_i$ and $EAP_i | EBP_i$ sequences defined in DFCv1

² These constants replace the KA_i and KB_i sequences defined in DFCv1

for $i = 1, 2, \dots, r$. Finally we define

$$\text{EK} = \text{EF}(K) = \text{RK}_1 | \text{RK}_2 | \dots | \text{RK}_r. \quad (12)$$

We can start the same process from $\text{IRK}_r | \text{RK}_r$ instead of PK . This enables to decrypt by computing the reversed sequence RK_i “on the fly”.

This new key schedule repairs two drawbacks which were reported on DFCv1 (see [2]). Namely, due to the pairwise difference of the IRK_i s, the iterations of the IEK_i s are no longer symmetric which fixes the weak key property reported by Coppersmith, and the first round key RK_1 now depends on all key bits. In addition, the RK_i sequence now looks “more random”.

2.6 On the Definition of the Constants

The previously defined algorithm depends on several constants:

- 64 constants $\text{RT}(0), \dots, \text{RT}(63)$ of $\frac{m}{4}$ bits (thus forming $16m$ bits),
- one $\frac{m}{2}$ -bit constant KD ,
- one $\frac{m}{4}$ -bit constant KC ,
- 16 m -bit constants $\text{KAB}_0, \dots, \text{KAB}_{15}$
- one $2m$ -bit constant KS .

Those constants must satisfy the following security criterion.

1. the RT round table has no collision,
2. KD is odd,
3. the IRK_j are pairwise different for $j = 1, \dots, rs$ ³.

We will use some constants several times. Actually, the RT table, KC and KD will contain the other constants. We thus need $18m$ bits of random constants.

In order to convince that this design hides no trap-door, we choose the constants from the hexadecimal expansion of the mathematical e constant

$$e = \sum_{n=0}^{\infty} \frac{1}{n!} = 2.\text{b7e151628aed2a6abf7158}_x \dots \quad (13)$$

We use the following scheme in order to define the constants.

Step 1. Let EES (as for “ e Expansion String”) be the first $18m$ bits of the expansion of e after the (hexa)decimal point, we define

$$\text{trunc}_{\frac{67}{4}m}(\text{EES}) = \text{RT}(0) | \text{RT}(1) | \dots | \text{RT}(63) | \text{KD} | \text{KC}. \quad (14)$$

³ Note that when this criterion is satisfied for one key, it is satisfied for any key

Here is the EES string for $m = 128$.

```
b7e15162 8aed2a6a bf715880 9cf4f3c7 62e7160f 38b4da56_x
a784d904 5190cfef 324e7738 926cfbe5 f4bf8d8d 8c31d763_x
da06c80a bb1185eb 4f7c7b57 57f59584 90cf4d7d 7c19bb42_x
158d9554 f7b46bce d55c4d79 fd5f24d6 613c31c3 839a2ddf_x
8a9a276b cfbfa1c8 77c56284 dab79cd4 c2b3293d 20e9e5ea_x
f02ac60a cc93ed87 4422a52e cb238fee e5ab6add 835fd1a0_x
753d0a8f 78e537d2 b95bb79d 8dcaec64 2c1e9f23 b829b5c2_x
780bf387 37df8bb3 00d01334 a0d0bd86 45cbfa73 a6160ffe_x
393c48cb bbca060f 0ff8ec6d 31beb5cc eed7f2f0 bb088017_x
163bc60d f45a0ecb 1bcd289b 06cbbfea 21ad08e1 847f3f73_x
78d56ced 94640d6e f0d3d37b e67008e1 86d1bf27 5b9b241d_x
eb64749a 47dfdfb9 6632c3eb 061b6472 bbf84c26 144e49c2_x
```

Step 2. We use the following algorithm to enforce the first two security criteria.

1. for $i = 0$ to 63 do
 - (a) while there exists $0 \leq j < i$ such that $RT(j) = RT(i)$, replace $RT(i)$ by $RT(i) + 1 \bmod 2^{\frac{m}{4}}$.
2. if KD is even, replace KD by $KD + 1$.
3. change the EES string accordingly so that Equation (14) holds.

Step 3. From this EES string we now define

$$EES = KAB_0 | \dots | KAB_{15} | KS. \quad (15)$$

Note that the third security criterion is necessarily satisfied, otherwise we would have collisions in RT.

At the end of the algorithm, we obtain a constant EES string depending on the parameters and which comes from the expansion of e and all the defined constants. We notice that for $m = 128$ all criteria are satisfied when EES is equal to the original expansion string of e (written in hexadecimal as above). For large m , it is highly unusual that we have to change it (but for KD with probability $1/2$).

3 Benchmarks and Implementations

Straightforward implementations of DFC are quite slow on 32-bit microprocessors for the nominal choices of parameters, due to the critical operation $ax + b \bmod 2^{64} + 13$. Efficient implementations require non trivial tricks. That is why the original implementation of DFCv1, which was bound to NISTs requirements (namely, ANSI-C implementation, which restricts to 32-bit words and prohibits the use of the 32-bit *times* 32-bit \rightarrow 64-bit multiplication of most processors), was quite slow and actually slower than most other candidates, especially since it dealt with endianness as well. The ANSI-C implementation

required 3600 clock cycles per encryption (without key setup) on a Pentium Pro. This should be compared with the 392 clock cycles on the same processor using assembly language and processor specific tricks. Further implementation tricks (which were summarized by Noilhan [15]) and clever use of specific architectures of microprocessors have shown that DFC was among the fastest AES candidates, and notably the fastest one on ALPHA 64-bit microprocessors (310 clock cycles per encryption without the key setup, on an ALPHA 21164a in assembly code⁴).

DFCv2 does not introduce important implementation differences from DFCv1 for the nominal choice of the parameters. More precisely, only the key schedule has changed, and even the complexity of the key setup has not changed (it roughly takes four basic encryptions).

4 Security Analysis

4.1 Provable Security Results

We state the security results in terms of the new parameters (m, k, r, s) .

Ideal key schedule. We recall that the security results consist, firstly of theoretical results for an ideal extension of DFCv2 in which the RK_i sequence is assumed to be uniformly distributed (we will call $DFCv2^*(m, r)$ this ideal algorithm which does not depend on k or s), secondly of some practical results on the real DFCv2 algorithm in which we have to make a heuristic assumption stated below.

Theorem 1 ([24]). *The best advantage of an attack limited to two adaptively chosen plaintexts for distinguishing $DFCv2^*(m, r)$ from a uniformly distributed random permutation is bounded by*

$$\text{BestAdv}_{\text{Cl}_a^2}(\text{DFCv2}^*(m, r), C^*) \leq \frac{1}{2} \left(3 \left(\left(\frac{p}{2^{\frac{m}{2}}} \right)^2 - 1 \right) + \frac{8}{2^{\frac{m}{2}}} \right)^{\lfloor \frac{r}{3} \rfloor} \quad (16)$$

where p is the smallest prime number greater than $2^{\frac{m}{2}}$.

If we let $p = 2^{\frac{m}{2}}(1 + \delta)$, the previous upper bound can be approximated by

$$\frac{1}{2} (6\delta + 2^{3-\frac{m}{2}})^{\lfloor \frac{r}{3} \rfloor}. \quad (17)$$

This shows that the best advantage is negligible against 2^{-m} if $r \geq 9$ when the attack is limited to two chosen plaintexts (i.e. in the $d = 2$ case). For $m = 128$, we have $\delta = 13.2^{-64}$ and we get back the bound of DFCv1

$$\text{BestAdv}_{\text{Cl}_a^2}(\text{DFCv2}^*(128, r), C^*) \leq \frac{1}{2} 2^{-57.5 \lfloor \frac{r}{3} \rfloor} \quad (18)$$

⁴ Implementation due to Robert Harley, see [8]. See also [1,15]

From the decorrelation theory we know that the security against any attack limited to two chosen plaintexts implies the security against some reasonable formalization of differential and linear cryptanalysis (see [20]). Namely, the average complexity of differential cryptanalysis (over the distribution of the keys) needs at least to be within the order of $1/4\text{BestAdv}$, as for the linear cryptanalysis (from an asymptotic bound). In this context, for instance, differential cryptanalysis can be formalized into:

1. pick a differential characteristic (a, b)
2. query an input pair of difference a until the corresponding output pair has a difference of b

It is well-known that this formalization is the core of regular differential cryptanalysis [3]. For instance, 2R attacks apply such a procedure on $r - 2$ rounds. Since we can claim that the differential cryptanalysis core against $\text{DFCv2}^*(128, 6)$ has a complexity of 2^{115} , we can thus claim that $\text{DFCv2}^*(128, 8)$ is secure against a 2R differential cryptanalysis up to a complexity of 2^{115} .

Similarly, the average complexity of any known plaintext coming from an iterated attack of order one (*i.e.* an iterated attack in which each iteration extracts one bit of information from one known plaintext/ciphertext pair) needs to be at least within the order of $1/2\sqrt{\text{BestAdv}}$ (see [22]).

More precisely, we recall the following result:

Theorem 2 ([20,22]). *For any differential distinguisher of complexity n against $\text{DFCv2}^*(m, r)$, the advantage Adv_D is such that*

$$\text{Adv}_D \leq n\text{BestAdv} + \frac{n}{2^m - 1} \quad (19)$$

where BestAdv is bounded by Equation (16). Similarly, for any linear distinguisher we have

$$\lim_{n \rightarrow +\infty} \frac{\text{Adv}_L}{n^{\frac{1}{3}}} \leq 9.3 \left(4\text{BestAdv} + \frac{1}{2^m - 1} \right)^{\frac{1}{3}}. \quad (20)$$

For any known plaintext iterated distinguisher of order 1 we have

$$\text{Adv}_I \leq 3 \left(\left(\frac{9}{2} 2^{-m} + 3\text{BestAdv} \right) n^2 \right)^{\frac{1}{3}} + n\text{BestAdv}. \quad (21)$$

Real key schedule. Since DFCv2 has a new key scheduling algorithm, we need to transform the security results on DFCv2^* to DFCv2. Let $\mathcal{D}(m, k, r, s)$ be the distribution of $(\text{RK}_1, \dots, \text{RK}_r)$ spanned by the key scheduling algorithm of $\text{DFCv2}(m, k, r, s)$ when K is a uniformly distributed k -bit key, and we let \mathcal{D}^* denote the uniform distribution over rm -bit sequences. DFCv2^* relies on the \mathcal{D}^* distribution, but DFCv2 uses the \mathcal{D} distribution.

Let $H_t(m, k, r, s)$ be the best advantage of a Turing machine limited to t steps for distinguishing $\mathcal{D}(m, k, r, s)$ from \mathcal{D}^* from a single sample (*i.e.* an rm -bit string). (H_t is a heuristic function. We need to assume that for a reasonable t , H_t is small.)

Theorem 3. *If for some class $Cl_{t,n}$ of distinguishers limited to a complexity of t and n oracle calls, the advantage for distinguishing $DFCv2^*(m, r)$ from a random permutation is limited to $BestAdv$, then the advantage for distinguishing $DFCv2(m, k, r, s)$ from a random permutation in class Cl is limited to $H_{t+O(n)}(m, k, r, s) + BestAdv$ where the $O(n)$ corresponds to the cost of simulating $DFCv2$ on n oracle calls.*

Therefore, assuming that the complexity of a practical attack already includes an overestimated cost for simulating the oracle calls (in practice, using an oracle costs more than simulating it), then all security results on $DFCv2^*$ extend to $DFCv2$ with an advantage offset of H_t .

For practical t , $m \geq 128$, $k \geq 128$, $s \geq 4$ and $r \leq \frac{128}{s}$, we conjecture that $H_t(m, k, r, s)$ is negligible.

4.2 Best Attacks

So far, the best reported attack is Knudsen's impossible differential attack [9] against $DFCv2$ reduced to six rounds. It requires 2^{70} chosen plaintexts and a complexity of 2^{126} encryptions (see [10]). This attack can be compared to a 1R attack that uses a differential characteristic on 5 rounds (for which the complexity lower bound indicated by Theorem 2 is of order 2^{57} chosen plaintexts).

Harvey recently reported⁵ an attack against four rounds which uses the non-injective properties of the round functions.

Another quite strong claim of insecurity is due to Rijmen and Knudsen [10]. Basically, they study a key-dependent one-round differential characteristic for a modified version of DFC and deduce some insecurity claims. One problem is that they use a difference which is not defined by the XOR operation but by the mod $2^{\frac{m}{2}}$ difference at the input and by the mod p difference at the output. This makes it hard to pile up such kinds of characteristics.

For instance, Rijmen and Knudsen noticed that if we replace all XORs in the round function by regular additions, every single input difference leads to about 800 possible output differences, one of it with probability 2^{-7} (with $m = 128$). These mod $2^{\frac{m}{2}}$ output differences translate into XOR output differences within a probability related to their Hamming weight (because of carry bits). We can thus estimate that the real DFC round function will lead to no key-dependent differential probabilities greater than 2^{-23} . Therefore, we believe the Rijmen-Knudsen observation does not imply any insecurity statement for $DFCv2$.

5 The DFC Controversy

The submission of $DFCv1$ to AES led to a controversy which was oriented towards three arguments which are addressed in the following subsections.

⁵ at the Rump Session of Fast Software Encryption 2000

5.1 Speed

DFCv1 was claimed to be among the slowest of the 15 AES candidates, and one of the worst for low-cost smart card implementations.

A fair performance comparison is a really hard task, as was shown by the AES conferences [18, section 4]. Timings have been collected by Granboulan [8] and Lipmaa [12], and DFC is without any doubt among the 8 fastest candidates in software: Crypton, DFC, E2, Mars, RC6, Rijndael, Serpent and Twofish. It is even the fastest candidate on architecture that have fast multiplication (Alpha and TurboSparc). When compared to the five finalists, DFC can be considered as achieving the same performances as Mars on current architectures (but being twice as fast on future architectures like Itanium). The dependence of DFC on multiplication can be compared to the dependence of RC6 on data dependent rotations.

In addition, it was shown in [17] that DFC was reasonably implementable on very simple embedded microprocessors (such as Motorola 6805 for smart cards). DFC does not take as much room on low-cost smart cards as Mars, and should have similar performances. On high-end smart cards (StrongARM) DFC is probably the fastest of all AES candidates.

In conclusion, DFC performances are not the best, but they compare very well to Mars, which is one of the finalists.

5.2 Provable Security

The provable security results were subject to controversy. We believe this was due to misunderstanding and we would like to clarify the situation.

After the DES was proposed, several other block ciphers showed up without any formal security argument. The security was essentially empirical: a block cipher was secure until someone came up with an attack. Although this approach proved very fruitful for promoting research on the analysis of block ciphers, the security provided is now debatable since the analysis time of all world experts is rather limited. Besides, we note that there were 15 candidates to analyze in less than one year, while DES weaknesses were discovered only after 10 years of public exposure.

Another tremendous amount of regular block ciphers use regular “security claims”, which essentially consists of heuristic arguments (like the argument on H_t we used above for DFCv2). Typically, people argue that we cannot get good differential characteristics by regular active S-box counting arguments. This paradigm was inherited by the work of Biham and Shamir [3] and Coppersmith’s analysis of DES [4].

In 1992, Lai and Massey [11] proposed the formal notion of “Markov cipher” which characterizes ciphers for which differentials can nicely be piled up. For these ciphers we can formally prove the heuristic security arguments against differential cryptanalysis on average over the key space.

Another more formal approach on which seldom block ciphers are based (including MISTY [13,14]) is inherited by Nyberg-Knudsen Theorem [16]. It

consists of using ad hoc constructions with heavy non-linear constraints on S-boxes and deducing that the block cipher has no good differential property on average on the key distribution. These results are however limited to differential (and linear) attacks.

Our paradigm obtains similar results to the previous approach in a more general setting for basically no cost. It further provides more freedom in the construction of the block cipher. Thus, we believe it is a better alternative which follows the construction trends.

One objection by Rijmen and Knudsen [10] argued that since there exist insecure algorithms for which similar security claims hold, such claims are worthless. Indeed, the affine cipher $x \mapsto K_1x + K_2$ has a perfect pairwise decorrelation, which means that Theorem 2 holds with $\text{BestAdv} = 0$, and in particular, no differential distinguisher gets a relevant advantage. (The differential is chosen before the attack itself in this model, so it is independent on the key.) This comes from the fact that we can “only” say that the probability of any differential is low on average over the key space. Previous formal approaches suffer from the same drawbacks. Actually, the Markov cipher approach is quite similar, and the Nyberg-Knudsen approach has the same result. As compared to the Nyberg-Knudsen approach, the present one holds for regular ciphers (not only to ad hoc constructions). Therefore we claim that DFCv2 benefits from the all regular heuristic security arguments and the present formal security proof (which is not the case of the affine cipher, nor of any other regular cipher). This suggests that DFC has its *raison d’être*.

5.3 Security Margin

Another criticism against DFC was its low “security margin”. The DFC philosophy consisted of not overestimating the minimal number of secure rounds and committing to the formal results obtained by decorrelation theory. We actually believe that for construction reasons, the security increases faster with the number of rounds than for other designs. We chose $r = 8$ as a challenge to the cryptographic community. Users who would not like to commit on such a bet can however freely use a higher number of rounds in the present DFCv2 version (for instance, $r = 12$ as recommended by Biham).

6 Conclusion

We have presented an updated version of DFC in which we changed the key schedule and introduced scalable parameters. These modifications left the security results unchanged (except the weak key attack which has been fixed).

Despite of the controversy during the AES process, we have shown that DFCv2 is one of the fastest block ciphers (on 64-bit microprocessors which have an optimized multiplier for $m = 128$) and benefits from some formal security results in addition to regular heuristic arguments.

Although this first generation of decorrelated ciphers may still be improved by the research community, we hope this paradigm will be useful to develop future cryptographic algorithms.

Acknowledgements

It was our pleasure to participate to the DFC team. For this we are deeply grateful to all other team members: Olivier Baudron, Henri Gilbert, Marc Girault, Helena Handschuh, Philippe Hoogvorst, Antoine Joux, David Pointcheval, Thomas Pornin, Guillaume Poupard and Jacques Stern, and particularly to Robert Harley who got interest in optimizing DFC implementations and who joined us.

We thank the CNRS, the École Normale Supérieure and France Telecom for sponsoring the project, particularly FIST (France Innovation Scientifique et Transfert), Jacques Stern and Henri Gilbert for devoting time to the initiative.

We wish to thank all the researchers who have got interest in DFC, particularly Don Coppersmith and Ron Rivest, and Dominik Behr, Brian Gladman, Danjel McGougan, Terje Mathisen, David Seal for their good optimizations.

We would finally like to thank NIST for having initiated this story and for having acknowledged us to participate.

A Test Vector

A test vector for the nominal choice of parameters ($m = 128$, $k \in \{128, 192, 256\}$, $r = 8$ and $s = 4$) is included below.

We have chosen to use KS as key and 0_x as plaintext. We recall the value of KS:

```
86d1bf27 5b9b241d eb64749a 47dfdfb9x
6632c3eb 061b6472 bbf84c26 144e49c2x
```

The key schedule tests all KAB entries but KAB_1 and KAB_{12} (which are not used with this choice of parameters). It results in the following subkeys:

| <i>round</i> | <i>subkeys</i> |
|--------------|--|
| 1 | 05c5bd24 aa6ba7df 0846cb21 e1ab0dc7 _x |
| 2 | 63b67a97 142061ce c034fd75 ea2cd3d9 _x |
| 3 | abf20d20 9b963b4c f04efdd6 2a6c459d _x |
| 4 | 27215d71 2b28c6cb e2f472eb 288d47e8 _x |
| 5 | 02aae49f caf2ddf3 60405b1d d0d269a7 _x |
| 6 | 2a516cdc 6270af2b f3db8f26 c26ea9eb _x |
| 7 | 94d3b898 ccbca828 4f6af189 39230738 _x |
| 8 | 6c9d3c7e d7059bcc 7a3d4288 f232b634 _x |

The iterated encryptions of plaintext 0_x tests all entries in the RT table for $j = 64$.

| j | DFCv2 $_KS^j$ |
|-----|--|
| 0 | 00000000 00000000 00000000 00000000 _x |
| 1 | 1ba5af95 aba096ed 5b6c9750 2fe7efa2 _x |
| 2 | 0f36105c 1302d52a e47d6d42 dfaaf5c7 _x |
| 3 | bb58f671 54c59d52 fefb03a8 74c138c5 _x |
| 4 | acc4cf76 6505c09f 5ffe10d5 b021d66c _x |
| 8 | 62395cc6 ba7bf158 f78b5897 04a1db59 _x |
| 16 | 387c4222 c61f5e69 7946e251 eb40031a _x |
| 32 | 4ab38d66 16247c2a efbe6cde 4d302a86 _x |
| 64 | ee043b7d a8610c46 3e282198 c93887b4 _x |

References

1. O. Baudron, H. Gilbert, L. Granboulan, H. Handschuh, A. Joux, P. Nguyen, F. Noilhan, D. Pointcheval, T. Pornin, G. Poupard, J. Stern, S. Vaudenay. Report on the AES Candidates. In *Proceedings from the Second Advanced Encryption Standard Candidate Conference*, National Institute of Standards and Technology (NIST), Rome, Italy, March 1999.
2. O. Baudron, H. Gilbert, L. Granboulan, H. Handschuh, R. Harley, A. Joux, P. Nguyen, F. Noilhan, D. Pointcheval, T. Pornin, G. Poupard, J. Stern, S. Vaudenay. DFC Update. In *Proceedings from the Second Advanced Encryption Standard Candidate Conference*, National Institute of Standards and Technology (NIST), Rome, Italy, March 1999.
3. E. Biham, A. Shamir. *Differential Cryptanalysis of the Data Encryption Standard*, Springer-Verlag, 1993.
4. D. Coppersmith. The Data Encryption Standard (DES) and its Strength against Attacks. *IBM Journal of Research and Development*, vol. 38, pp. 243–250, 1994.
5. H. Feistel. Cryptography and computer privacy. *Scientific American*, vol. 228, pp. 15–23, 1973.
6. H. Gilbert, M. Girault, P. Hoogvorst, F. Noilhan, T. Pornin, G. Poupard, J. Stern, S. Vaudenay. Decorrelated Fast Cipher: an AES Candidate. (Extended Abstract.) In *Proceedings from the First Advanced Encryption Standard Candidate Conference*, National Institute of Standards and Technology (NIST), Ventura, California, U.S.A., August 1998.
7. H. Gilbert, M. Girault, P. Hoogvorst, F. Noilhan, T. Pornin, G. Poupard, J. Stern, S. Vaudenay. Decorrelated Fast Cipher: an AES Candidate. Submitted to the Advanced Encryption Standard process. In *CD-ROM “AES CD-1: Documentation”*, National Institute of Standards and Technology (NIST), August 1998.
8. L. Granboulan. AES ; Timings of the best known implementations. <http://www.di.ens.fr/~granboul/recherche/AES/timings.html>
9. L. R. Knudsen. DEAL - A 128-Bit Block Cipher. Submitted to the Advanced Encryption Standard process. In *CD-ROM “AES CD-1: Documentation”*, National Institute of Standards and Technology (NIST), August 1998.

10. L. R. Knudsen, V. Rijmen. On the Decorrelated Fast Cipher (DFC) and Its Theory. In *Fast Software Encryption*, Roma, Italy, Lectures Notes in Computer Science 1636, pp. 81–94, Springer-Verlag, 1999.
11. X. Lai. *On the Design and Security of Block Ciphers*, ETH Series in Information Processing, vol. 1, Hartung-Gorre Verlag Konstanz, 1992.
12. H. Lipmaa. AES Ciphers: speed <http://www.tcm.hut.fi/~helger/aes/>
13. M. Matsui. New Structure of Block Ciphers with Provable Security against Differential and Linear Cryptanalysis. In *Fast Software Encryption*, Cambridge, United Kingdom, Lectures Notes in Computer Science 1039, pp. 205–218, Springer-Verlag, 1996.
14. M. Matsui. New Block Encryption Algorithm MISTY. In *Fast Software Encryption*, Haifa, Israel, Lectures Notes in Computer Science 1267, pp. 54–68, Springer-Verlag, 1997.
15. F. Nollan. Software Optimization of Decorrelation Modules. In *Selected Areas in Cryptography*, Kingston, Ontario, Canada, Lectures Notes in Computer Science 1758, pp. 175–183, Springer-Verlag, 2000.
16. K. Nyberg, L. R. Knudsen. Provable Security against a Differential Cryptanalysis. *Journal of Cryptology*, vol. 8, pp. 27–37, 1995.
17. G. Poupard, S. Vaudenay. Decorrelated Fast Cipher: an AES Candidate well suited for Low Cost Smart Cards Applications. In *CARDIS' 98*, Louvain-la-Neuve, Belgium, Lectures Notes in Computer Science 1820, pp. 254–264, Springer-Verlag, 2000.
18. B. Preneel *et al.* Comments by the NESSIE Project on the AES Finalists. Submitted to the Advanced Encryption Standard process, round 2 comments. National Institute of Standards and Technology (NIST), May 2000. <http://csrc.nist.gov/encryption/aes/round2/comments/20000524-bpreneel.pdf>
19. R.L. Rivest. The RC5 Encryption Algorithm. In *Fast Software Encryption*, Leuven, Belgium, Lectures Notes in Computer Science 1008, pp. 86–96, Springer-Verlag, 1995.
20. S. Vaudenay. Provable Security for Block Ciphers by Decorrelation. Invited talk. In *STACS 98*, Paris, France, Lectures Notes in Computer Science 1373, pp. 249–275, Springer-Verlag, 1998. Full Paper: technical report LIENS-98-8, Ecole Normale Supérieure, 1998. (<ftp://ftp.ens.fr/pub/reports/liens/>)
21. S. Vaudenay. Feistel Ciphers with L_2 -Decorrelation. In *Selected Areas in Cryptography*, Kingston, Ontario, Canada, Lectures Notes in Computer Science 1556, pp. 1–14, Springer-Verlag, 1999.
22. S. Vaudenay. Resistance Against General Iterated Attacks. In *Advances in Cryptology EUROCRYPT'99*, Prague, Czech Republic, Lectures Notes in Computer Science 1592, pp. 255–271, Springer-Verlag, 1999.
23. S. Vaudenay. On the Lai-Massey Scheme.
24. S. Vaudenay. Adaptive-Attack Norm for Decorrelation and Super-Pseudorandomness. In *Selected Areas in Cryptography*, Kingston, Ontario, Canada, Lectures Notes in Computer Science 1758, pp. 49–61, Springer-Verlag, 2000.
25. S. Vaudenay. On Provable Security for Conventional Cryptography. Invited talk. (To appear in the proceedings of ICISC' 99, LNCS, Springer-Verlag.)

The Block Cipher Hierocrypt

Kenji Ohkuma¹, Hirofumi Muratani¹, Fumihiko Sano², and
Shinichi Kawamura¹

¹ Toshiba Corporate R & D Center

`kenji.ohkuma@toshiba.co.jp`

² Toshiba SI Thechnology Center

Abstract. This paper proposes a nested (hierarchical) SPN structure and the symmetric block cipher “Hierocrypt”. In the nested SPN structure, lower-level SPN structures are recursively embedded into S-box positions in SPN of the higher level. This structure recursively assures the lower bound of active S-box number, and high security level is efficiently realized. The 8-round Hierocrypt is implemented in C language on Pentium III, and shows the middle-class performance of final AES candidates.

1 Introduction

The substitution-permutation network (SPN, for short) is one of the most important structures besides the Feistel network. The wide trail strategy is effective for an SPN cipher to achieve high security against the differential and linear cryptanalysis[1,2,3].

The optimal invertible linear mapping of diffusion layer is an essential component for the wide trail strategy. The mapping is usually called the maximum distance separable (MDS) mapping [2,4,5]. MDS mapping is optimal for the number of active S-boxes, which ensures the upper-bound of the characteristic probability for differential and linear cryptanalysis.

Rijmen et al. designed the 64-bit block cipher SHARK, where eight-parallel 8-bit S-boxes is mixed by the permutation layer, and the number of active S-boxes in the two consecutive layers is at least 9 [2]. It seems that the structure of SHARK is effective for a larger block size. But, straightforward extension to a larger block size has a disadvantage that the calculational cost for MDS part is proportional to the square of block size¹.

As a solution to the problem, Daemen, Rijmen et al. proposed the 128-bit ciphers SQUARE and Rijndael, where sixteen 8-bit S-boxes are divided into four parts composed of four S-boxes, and a local MDS operation is applied to each of them [4,5]. Although the minimum number of active S-boxes in consecutive two rounds is only 5, any trail of four consecutive rounds has at least 25 active S-boxes.

¹ The MDS operation consists of matrix multiplication where the number of matrix elements is proportional to the square of matrix size

We propose a new class of SPN structure, a nested (or hierarchical) SPN structure, and the cipher Hierocrypt² based on the structure in this paper. The nested SPN structure is a multiple-level recursive structure (See Figure 1), and recursively assures the lower bound of active S-box number, and high security level is efficiently realized. The nested SPN structure can be regarded as an generalization of the SQUARE/Rijndael-type cipher. The generalization makes it possible to improve the security against the SQUARE attack.

The construction of this paper is as follows. In the following section, the nested SPN structure is introduced. In Sect. 3, we show an overview of the cipher Hierocrypt, which is composed of the nested SPN structure. In Sect. 4, we describe how the components of Hierocrypt are designed. In Sect. 5, the security of Hierocrypt against some attacks is discussed. Sect. 6 shows the software performance of 8-round Hierocrypt on some CPU. The final section is devoted to the concluding remarks.

2 Nested SPN Structure

A nested (hierarchical) SPN structure is a multiple-level structure, where an S-box in a certain level consists of a 1-level lower SPN (See Figure 1).

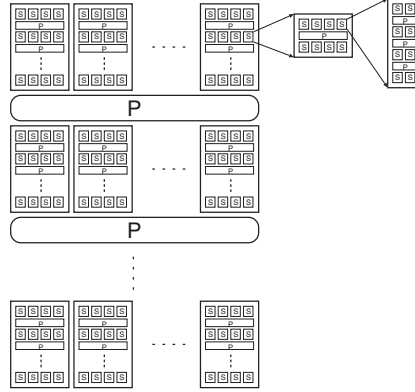


Fig. 1. Nested SPN structure

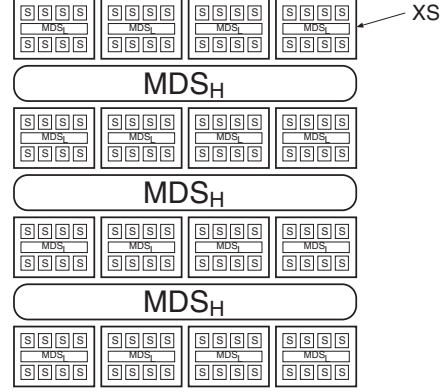


Fig. 2. 4-round nested SPN cipher

We impose the following conditions to realize the wide trail strategy efficiently [2,4,5].

- (a) The final round of SPN consists only of an S-box layer (not followed by a diffusion layer) in all levels;
- (b) All permutations are MDS in each level;
- (c) The number of rounds is even in all levels except for the highest;

² Two versions of the cipher (Type-I and Type-II) is proposed in this paper

- (d) Bit-wise key additions are located directly before all lowest-level S-box layers and directly after the final;

Figure 2 shows an example of two-level nested SPN cipher, which consists of 3 rounds of the higher-level SPN, whose S-boxes have a 2-round SPN structure. The number of parallel S-boxes is 4 for both levels. When the size of S-box is 8-bit, the block length of the cipher is 128-bit.

The following proposition is important to realize the wide trail strategy in the nested SPN.

Proposition 1. Consider 2-level nested SPN with the following conditions: (i) the above conditions (a)~(c) are satisfied; (ii) the lower-level SPN is 2-round; (iii) the number of parallel S-boxes are m_1 and m_2 for the higher- and lower-level SPN, respectively. Then, any 2 consecutive higher-level rounds contain no less than $(m_1 + 1)(m_2 + 1)$ lower-level active S-boxes, for nonzero differential/mask.

Proof. For non-zero differential or non-zero mask pattern, there are no less than $(m_1 + 1)$ active higher-level S-boxes, each of which has no less than $(m_2 + 1)$ active lower-level S-boxes. Thus, at least $(m_1 + 1)(m_2 + 1)$ lower-level S-boxes are active. \square

To construct a nested SPN cipher which satisfies the above condition (b), an MDS code with a large word-size is needed. Here, let the (n, k, d) code be a code where n is the block length, k is the number of information digits, and d is the minimum distance. For the case of Figure 2, we need (8,4,5) error-correcting code over 32-bit word set for the higher-level diffusion. Although the Reed-Solomon code over $\text{GF}(2^{32})$ satisfies the condition, calculation over $\text{GF}(2^{32})$ is often costly. An alternative way is construction by concatenating parallel smaller MDS-codes, which is based on the following proposition.

Proposition 2. Let MDS_n be an MDS mapping defined by a $(2m, m, m + 1)$ -code over the n -bit word set. Then an MDS mapping $MDS_{m'n}$ based on $(2m, m, m + 1)$ -code over $m'n$ -bit word set is constructed by concatenating m' sets of MDS_n .

Proof. The proposition is proven by constructing an example of $(2m, m, m + 1)$ -code over $m'n$ -bit words.

Consider m' sets of mapping MDS_n .

$$MDS_n : x_{1j} \| x_{2j} \| \cdots \| x_{mj} \mapsto y_{1j} \| y_{2j} \| \cdots \| y_{mj} , \quad 1 \leq j \leq m' ,$$

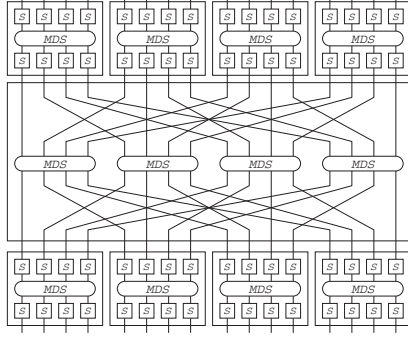
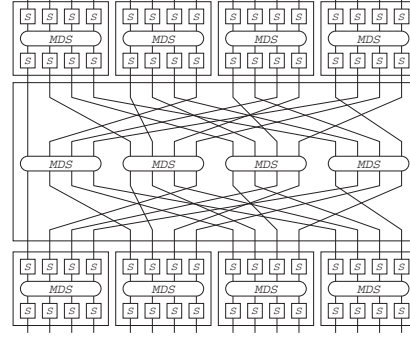
and define the concatenation as follows.

$$X_i = x_{i1} \| x_{i2} \| \cdots \| x_{im'} , \quad Y_i = y_{i1} \| y_{i2} \| \cdots \| y_{im'} , \quad 1 \leq i \leq m .$$

Then define the following mapping $MDS_{m'n}$

$$MDS_{m'n} : X_1 \| X_2 \| \cdots \| X_m \mapsto Y_1 \| Y_2 \| \cdots \| Y_m .$$

For nonzero differential/mask, at least one of the m' sub-mappings MDS_n is active, therefore at least $(m + 1)$ $m'n$ -bit words from $\{X_i\}$ and $\{Y_i\}$ are active. This means that the mapping is MDS. \square

**Fig. 3.** SQUARE in nested SPN form**Fig. 4.** Rijndael in nested SPN form

The above construction of $MDS_{m'n}$ is a fundamental one. The mapping is generalized by putting invertible linear transformations on input and output words of $m'n$ -bit length.

The above construction of MDS mapping gives a new viewpoint for SQUARE and Rijndael. Figures 3 and 4 respectively show mathematically equivalent forms of four-round SQUARE and Rijndael. All MDS are the same. The central large rectangle corresponds to the MDS of higher level, consisting of 4 parallel sub-MDS of (8, 4, 5)-code. Thus, the parameters are $m = m' = 4$; $n = 8$, which guarantees that no less than 25 S-boxes are active in four consecutive rounds.

That is, the same code is used in both-level MDS mapping for SQUARE.

3 Overview of Hierocrypt Encryption

Hierocrypt is a two-level nested SPN cipher with which has the following features.

- (α) The size of lower-level S-box is 8-bit;
- (β) The number of parallel S-boxes is 4 in both levels;
- (γ) Lower-level structure is 2-round SPN;
- (δ) Diffusion layers of both levels consist of MDS mapping defined by (8, 4, 5)-code.

The lower-level diffusion MDS_L is based on (8, 4, 5)-code over $GF(2^8)$. Two types of higher-level diffusions MDS_H are designed on (8, 4, 5)-codes over $GF(2^{32})$ and over $GF(2^4)$, respectively.

For respective Galois fields, the following primitive polynomials are used in this paper.

$$\begin{aligned} p_4(z) &= z^4 + z + 1, & \text{for } GF(2^4), \\ p_8(z) &= z^8 + z^6 + z^5 + z + 1, & \text{for } GF(2^8), \\ p_{32}(z) &= z^{32} + z^{28} + z^{27} + z + 1, & \text{for } GF(2^{32}). \end{aligned}$$

3.1 Lower-Level Components of Hierocrypt

The lower-level SPN structure of Hierocrypt consists of three components: bit-wise key addition AK , byte substitution $[S]$, and diffusion $[MDS_L]$.

Bit-Wise Key Addition AK (1-Bit \times 128). 128-bit half-round key K is added to the data (See Sect. 3.4 for the key scheduling).

$$AK(K) : X \mapsto Y \iff Y = X \oplus K .$$

Byte Substitution $[S]$ (8-Bit \times 16). The byte substitution $[S]$ consists of parallel operations of S-box S (See Sect. 4.1).

$$[S] : X \mapsto Y \iff y_{ij} = S(x_{ij}) , \quad i = 1, \dots, 4 ; j = 1, \dots, 4 .$$

The square bracket $[]$ means parallel operation.

Lower-Level Diffusion $[MDS_L]$ (32-Bit \times 4). The lower-level diffusion MDS_L mixes 4 parallel bytes (See Sect. 4.2).

$$[MDS_L] : X \mapsto Y \iff Y_i = MDS_L(X_i) , \quad i = 1, \dots, 4 .$$

3.2 Higher-Level Components of Hierocrypt

The higher-level SPN consists of two components: parallel 4-byte substitution $[XS]$ and higher-level diffusion MDS_H , except for the final key addition.

4-Byte Substitution $[XS]$ (32-Bit \times 4).

$$[XS] : X \mapsto Y \iff Y_i = XS(X_i) , \quad i = 1, \dots, 4 .$$

4-byte substitution $[XS]$ consists of the three kinds of lower-level components, combined as follows.

$$[XS] = [S] \circ AK(K^\beta) \circ [MDS_L] \circ [S] \circ AK(K^\alpha) ,$$

where K^α and K^β is the first and the second half of round key (See Sect. 3.4).

Higher-Level Diffusion MDS_H (128-Bit \times 1).

$$MDS_H : X \mapsto Y .$$

Two types are given in Sect. 4.3 and 4.3.

3.3 Round Functions and Encryption

Round Function ρ (Except for Final).

$$\rho = MDS_H \circ [XS] .$$

Final Round Function ρ' .

$$\rho' = [XS] .$$

Hierocrypt Encryption. The Hierocrypt encryption of T rounds consists of $(T-1)$ iterations of round function ρ followed by the final round function ρ' and the final key addition.

$$Enc = AK(K^{(T+1)\alpha}) \circ \rho'(K^{(T)}) \circ \rho(K^{(T-1)}) \circ \dots \circ \rho(K^{(2)}) \circ \rho(K^{(1)}) .$$

3.4 Key Scheduling

The key scheduling part consists of an initial key expansion KX and iterative key generations KH .

$$\begin{aligned} K^{(0)} &= KX(K) , \\ K^{(t)} &= KH(K^{(t-1)}) , \quad (1 \leq t \leq 2T+1) . \end{aligned}$$

The data randomization part requires two-round iterations of KH per round.

Initial Key Expansion. The initial key expansion KX expands an encryption key K (128/192/256 bits) up to 256-bit by padding. The 32-bit key data K_i is represented as concatenation of four 8-bit data.

$$K_i = k_{i1} \| k_{i2} \| k_{i3} \| k_{i4} .$$

[128-bit key]

$$\begin{aligned} K &= K_1 \| K_2 \| K_3 \| K_4 \\ &= k_{11} \| k_{12} \| k_{13} \| k_{14} \| k_{21} \| k_{22} \| \dots \| k_{43} \| k_{44} . \\ K^{(0)} &= K_1 \| K_2 \| K_3 \| K_4 \| K_1 \| K_2 \| K_3 \| K_4 . \end{aligned}$$

[192-bit key]

$$\begin{aligned} K &= K_1 \| K_2 \| K_3 \| K_4 \| K_5 \| K_6 \\ &= k_{11} \| k_{12} \| \dots \| k_{43} \| k_{44} \| k_{51} \| k_{52} \| \dots \| k_{63} \| k_{64} . \\ K^{(0)} &= K_1 \| K_2 \| K_3 \| K_4 \| K_5 \| K_6 \| K_1 \| K_2 . \end{aligned}$$

[256-bit key]

$$K^{(0)} = K = K_1 \| K_2 \| K_3 \| K_4 \| K_5 \| K_6 \| K_7 \| K_8 .$$

Key Round Function. The key round function transforms the $(t-1)$ -th intermediate key $K^{(t-1)}$ into the t -th one $K^{(t)}$,

$$K^{(t)} = KH(K^{(t-1)}) .$$

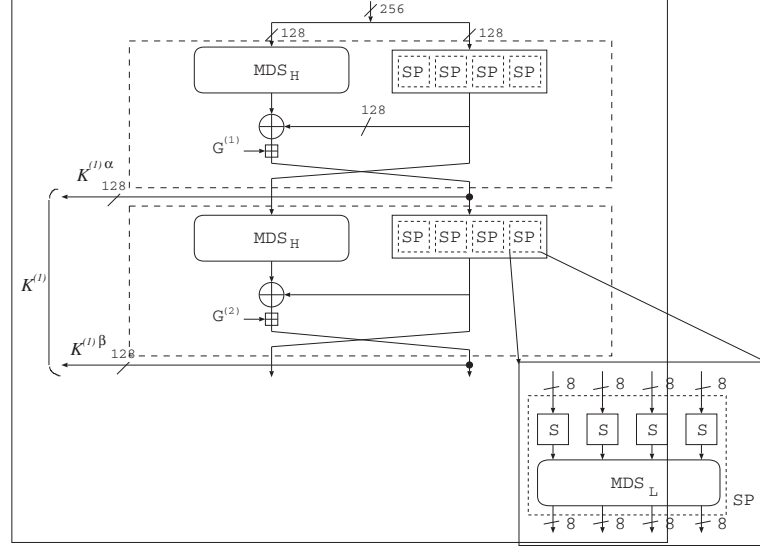


Fig. 5. Overview of the key scheduling

The key $K^{(t)}$ divides into two halves: $KL^{(t)}$ (left half) and $KR^{(t)}$ (right half),

$$K^{(t)} = KL^{(t)} \| KR^{(t)} .$$

The output of key round function corresponds to the round key for data randomization as follows.

$$K^{(t)\alpha} = KR^{(2t-1)} , \quad K^{(t)\beta} = KR^{(2t)} .$$

The outline of the key round function is as follows (See Fig.5).

- **[right \mapsto left]**

$$[MDS_L] \circ [S] : \quad KR^{(t-1)} \mapsto KL^{(t)} .$$

where $[S]$ is given in Sect. 3.1.2, and $[MDS_L]$ is given in Sect. 3.1.3.

- **[left \mapsto right]**

$$AD_+ \left(G^{(t)} \right) \circ AD \left(KL^{(t)} \right) \circ MDS_H : \quad KL^{(t-1)} \mapsto KR^{(t)} .$$

where MDS_H is given in Sect. 3.2.2, $AD_+ \left(G^{(t)} \right)$ is the addition of round constant $G^{(t)}$.

$$KR_i^{(t)} = KR_i^{(t-1)} + G_i^{(t)} \pmod{2^{32}} , \quad (i = 1, \dots, 4) .$$

The constants $G^{(t)}$ are given in Table 1.

Table 1. Round constant $G^{(t)}$

| | |
|------------|--------------------------|
| $G^{(1)}$ | $= (H_2, H_0, H_1, H_1)$ |
| $G^{(2)}$ | $= (H_3, H_2, H_0, H_3)$ |
| $G^{(3)}$ | $= (H_1, H_0, H_0, H_0)$ |
| $G^{(4)}$ | $= (H_1, H_0, H_1, H_3)$ |
| $G^{(5)}$ | $= (H_0, H_1, H_0, H_2)$ |
| $G^{(6)}$ | $= (H_3, H_2, H_0, H_0)$ |
| $G^{(7)}$ | $= (H_1, H_2, H_1, H_0)$ |
| $G^{(8)}$ | $= (H_2, H_1, H_2, H_3)$ |
| $G^{(9)}$ | $= (H_2, H_1, H_0, H_0)$ |
| $G^{(10)}$ | $= (H_1, H_1, H_1, H_2)$ |
| $G^{(11)}$ | $= (H_3, H_1, H_1, H_2)$ |
| $G^{(12)}$ | $= (H_1, H_1, H_2, H_0)$ |
| $G^{(13)}$ | $= (H_1, H_3, H_3, H_1)$ |
| $G^{(14)}$ | $= (H_2, H_3, H_3, H_1)$ |
| $G^{(15)}$ | $= (H_1, H_3, H_1, H_0)$ |
| $G^{(16)}$ | $= (H_1, H_0, H_0, H_3)$ |
| $G^{(17)}$ | $= (H_1, H_2, H_0, H_3)$ |

Generation of Round Constant $G^{(t)}$. To prevent the weak key generation such as a cyclic pattern in round key sequence, we introduce 32-bit constant parameters $G_i^{(t)} (t = 1, \dots, 17; i = 1, \dots, 4)$, which are given by Table 1. Here, the constants $H_i (i = 1, \dots, 4)$ is given as follows (The prefix “0x” indicates hexadecimal numbers).

$$H_0 = 0x5A827999 = \text{trunc}(\sqrt{2}/4), \quad H_1 = 0x6ED9EBA1 = \text{trunc}(\sqrt{3}/4),$$

$$H_2 = 0x8F1BBCDC = \text{trunc}(\sqrt{5}/4), \quad H_3 = 0xCA62C1D6 = \text{trunc}(\sqrt{10}/4),$$

where ‘*trunc*’ is the truncation function which is defined by using the floor function³.

$$\text{trunc}(x) = \lfloor 2^{32}x \rfloor.$$

Table 1 is given by the following simple rule. We use the eight-bit linear feedback shift register (LFSR) of Fibonacci type, of which the primitive polynomial is $z^8 + z^4 + z^3 + z^2 + 1$, and the initial state is $z^7 + z^3 + z + 1$. The LFSR generates a bit sequence $\zeta_1, \zeta_2, \zeta_3, \dots$. Successive two bits of them determines the suffix of H_j . Specifically, the i -th constant of the t -th round $G_i^{(t)}$ is given as follows.

$$G_i^{(t)} = H_{\phi(t,i)}, \quad \phi(t,i) = 2\zeta_{8(t-1)+2i-1} + \zeta_{8(t-1)+2i}.$$

4 Design of the Components

We describe how the components of Hierocrypt are designed in this section.

³ The floor function $\lfloor x \rfloor$ is the largest integer no more than x

In the component design, the maximum differential and linear probabilities are the most important security measures for block ciphers. The maximum differential probability for the function f is defined as

$$dp^f \equiv \max_{\Delta x \neq 0, \Delta y} \frac{\#\{x | f(x) \oplus f(x \oplus \Delta x) = \Delta y\}}{2^n}. \quad (1)$$

Similarly, the maximum linear probability for the function f is given as

$$lp^f \equiv \max_{\Gamma x, \Gamma y \neq 0} \left| \frac{\#\{x | x \cdot \Gamma x = f(x) \cdot \Gamma y\}}{2^n} - \frac{1}{2} \right|. \quad (2)$$

4.1 Lower-Level S-Box S

The lower-level S-box S is given as follows (in hexadecimal expression).

$$\begin{aligned} & (S(0) \ S(1) \ \cdots \ S(F) \ S(10) \ \cdots \ S(FF)) \\ &= (\begin{array}{l} 72 \ 4A \ 49 \ 16 \ 1E \ 3A \ 43 \ AE \ 66 \ BC \ 00 \ 73 \ 79 \ 3B \ FB \ 9F \\ 69 \ 6A \ A2 \ 50 \ 6E \ F5 \ EF \ AC \ 22 \ 02 \ AD \ 26 \ E2 \ DF \ 97 \ F0 \\ 9E \ BF \ 17 \ 8B \ FA \ 7C \ F4 \ 71 \ 7F \ CA \ F6 \ 52 \ FD \ C3 \ E5 \ 64 \\ 53 \ 8D \ E0 \ F3 \ 0F \ 78 \ CB \ 9B \ 68 \ 3C \ 0D \ 1F \ 89 \ B6 \ EB \ F7 \\ 44 \ 4A \ 06 \ A6 \ 56 \ 6B \ 85 \ 01 \ 30 \ 88 \ 51 \ 31 \ 9C \ A0 \ A3 \ 25 \\ 60 \ 5B \ FF \ 05 \ B7 \ 91 \ 15 \ B3 \ A9 \ 20 \ 03 \ 2B \ 61 \ 42 \ 95 \ 4D \\ F9 \ 7E \ 0E \ E9 \ D8 \ F1 \ 46 \ 99 \ CEBE \ D9 \ 54 \ 80 \ B0 \ D2 \ 4F \\ 7A \ E8 \ 35 \ 92 \ 1B \ 7B \ 12 \ D6 \ 4C \ D5 \ E7 \ EE \ B1 \ 24 \ DE \ 21 \\ 04 \ 10 \ AB \ 29 \ 9A \ 81 \ FE \ A7 \ B8 \ 63 \ 28 \ 0A \ 8A \ D1 \ C6 \ 07 \\ B9 \ C8 \ 98 \ 82 \ 74 \ 9D \ 84 \ 47 \ 94 \ C7 \ 6C \ 11 \ D7 \ BA \ C1 \ C9 \\ DD \ 77 \ 39 \ 2F \ 2E \ C2 \ 67 \ 41 \ E4 \ 58 \ 34 \ CD \ 1C \ 93 \ 96 \ 7D \\ 2C \ F8 \ B5 \ 70 \ 14 \ 08 \ DCCC \ 87 \ D0 \ 5E \ 32 \ C5 \ C4 \ 59 \ 3E \\ CF \ 55 \ 5C \ 23 \ 75 \ 2D \ 2A \ 86 \ 4B \ 1D \ 5F \ E6 \ FC \ B2 \ 4E \ 09 \\ 27 \ AF \ 19 \ B4 \ BD \ 6D \ 3D \ 6F \ ED \ 62 \ EA \ F2 \ D3 \ 36 \ 38 \ DB \\ BB \ 83 \ 45 \ 37 \ A4 \ EC \ 8C \ 5D \ E1 \ 33 \ 90 \ A1 \ 40 \ 8E \ 1A \ A5 \\ 0B \ 3F \ 5A \ DA \ 13 \ 76 \ 0C \ C0 \ 48 \ E3 \ 65 \ A8 \ 18 \ 8F \ D4 \ 57 \end{array}). \end{aligned}$$

The maximum differential probability of the lower-level S-box S is

$$dp^S = \frac{6}{256}.$$

And the distribution of differential probabilities for nonzero input differentials is shown in Table 2.

Similarly, for the maximum linear probability,

$$lp^S = \frac{22}{256}.$$

And the distribution of the linear probabilities for nonzero output mask patterns is shown in Table 3.

The algebraic order of the S-box is 7-th, which is the highest value for 8-bit bijection.

Table 2. Differential probability distribution

| differential probability (/256) | num. |
|---------------------------------|--------|
| 6 | 50 |
| 4 | 1,899 |
| 2 | 28,692 |
| 0 | 34,639 |
| total | 65,280 |

Table 3. Linear probability distribution

| linear probability (/256) | num. |
|---------------------------|----------|
| 22 | 20 |
| 20 | 157 |
| 18 | 577 |
| 36 | 579 |
| \vdots | \vdots |
| 0 | 5,748 |
| total | 65,280 |

4.2 Lower-Level Diffusion MDS_L and Higher-Level S-Box XS

The security of higher-level S-box XS depends on the combination of S-box S and permutation MDS_L . We have chosen MDS_L from randomly generated MDS matrices over $GF(2^8)$, such that the differential and linear properties are better than the usual case where all active S-boxes take the worst probability.

As the branch number of MDS_L is 5, the maximum differential and linear characteristic probabilities: DP^{XS} and LP^{XS} , respectively satisfy the following inequalities.

$$DP^{XS} \leq \left(\frac{6}{256} \right)^5 \cong 2^{-27.1} ,$$

$$LP^{XS} \leq \frac{1}{2} \left(2 \cdot \frac{22}{256} \right)^5 \cong 2^{-13.7} .$$

The above inequalities are satisfied, only if the lower-level permutation MDS_L is an MDS mapping. We succeeded in improving the differential and linear properties by selecting the following MDS_L .

$$MDS_L(X_i) = D_L X_i ,$$

$$D_L = \begin{pmatrix} 6C & 25 & 9B & 03 \\ 6D & 06 & C8 & 18 \\ 75 & 78 & 9E & 1F \\ 42 & 78 & EB & 61 \end{pmatrix} .$$

The matrix elements are expressed in hexadecimal and regarded as elements of $GF(2^8)$. For example, the elements 25 is regarded as the polynomial $z^5 + z^2 + 1$ of $GF(2^8)$.

$$\begin{aligned}
0x25 &= 1 \cdot 2^5 + 0 \cdot 2^4 + 0 \cdot 2^3 + 1 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0 \\
&\iff 1 \cdot z^5 + 0 \cdot z^4 + 0 \cdot z^3 + 1 \cdot z^2 + 0 \cdot z^1 + 1 \cdot z^0 = z^5 + z^2 + 1 .
\end{aligned}$$

We use the following criteria for selection.

- At least one S-box has a differential probability no more than $(4/256)$, for any nonzero input differential of which active S-box number is 5 (See Table 2).
- At least one S-box has a linear probability no more than $(20/256)$, for any nonzero output mask pattern of which active S-box number is 5 (See Table 3).

If MDS_L satisfies the criteria, the inequalities for XS are refined as follows.

$$\begin{aligned}
DP^{XS} &\leq \frac{4}{256} \left(\frac{6}{256} \right)^4 \cong 2^{-27.7} , \\
LP^{XS} &\leq \frac{1}{2} \left(2 \cdot \frac{20}{256} \right) \left(2 \cdot \frac{22}{256} \right)^4 \cong 2^{-13.8} .
\end{aligned}$$

4.3 Higher-Level Diffusion MDS_H

The higher-level diffusion MDS_H is based on $(8, 4, 5)$ -code over 32-bit words. We give two examples, [Type-I]: mapping based on $(8, 4, 5)$ -code over $GF(2^{32})$; [Type-II]: concatenation of 8 parallel mappings based on $(8, 4, 5)$ -code over $GF(2^4)$.

Type-I MDS_H . Type-I MDS_H is selected from randomly generated matrices over $GF(2^{32})$. For calculational efficiency, we impose the condition that only lowest 5 bits can be nonzero for all matrix elements. Multiplication with a constant over $GF(2^{32})$ reduces to 4 times of table-lookup where the respective inputs are 8-bit long.

$$\begin{aligned}
MDS_H(X) &= D_H X , \\
D_H &= \begin{pmatrix} 05 & 19 & 06 & 1B \\ 1B & 05 & 19 & 06 \\ 06 & 1B & 05 & 19 \\ 19 & 06 & 1B & 05 \end{pmatrix} .
\end{aligned}$$

Type-II MDS_H . Type-II MDS_H is based on the following 4×4 matrix D_h , which is selected from randomly generated matrices over $GF(2^4)$.

$$\begin{aligned}
MDS_H(X) &= D_H X , \quad D_H = D_h \otimes I_8 , \\
D_h &= \begin{pmatrix} 6 & B & D & C \\ C & 6 & B & D \\ D & C & 6 & B \\ B & D & C & 6 \end{pmatrix}
\end{aligned}$$

Here, I_8 means 8-dimensional identity matrix. $D_h \otimes I_8$ is eight parallel multiplications of the matrix D_h to the 16-bit vector (regarded as four elements of $\text{GF}(2^4)$) which is made by picking up one bit from each byte.

The byte-oriented form of mapping is as follows.

$$\begin{pmatrix} y_{11} \\ y_{12} \\ y_{13} \\ y_{14} \\ y_{21} \\ y_{22} \\ y_{23} \\ y_{24} \\ y_{31} \\ y_{32} \\ y_{33} \\ y_{34} \\ y_{41} \\ y_{42} \\ y_{43} \\ y_{44} \end{pmatrix} = \begin{pmatrix} 0 & 1 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 1 & 1 & 0 & 1 \\ 0 & 1 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 0 \end{pmatrix} \begin{pmatrix} x_{11} \\ x_{12} \\ x_{13} \\ x_{14} \\ x_{21} \\ x_{22} \\ x_{23} \\ x_{24} \\ x_{31} \\ x_{32} \\ x_{33} \\ x_{34} \\ x_{41} \\ x_{42} \\ x_{43} \\ x_{44} \end{pmatrix}$$

5 Security

5.1 Differential and Linear Cryptanalysis

The branch number is 5 for both mappings MDS_H and MDS_L . Thus, Hierocrypt contains at least 5 active XS per two rounds, which contain at least 5 active S-boxes, for a non-zero differential/mask. Therefore, Hierocrypt contains at least 25 active S-boxes per two rounds (See Proposition 1).

Then, the maximum differential and linear characteristics respectively satisfy the following inequalities.

$$DP^{2\text{ rounds}} = \left(\frac{6}{256} \right)^{25} \cong 2^{-135.3} \ll 2^{-128} .$$

$$LP^{2\text{ rounds}} \leq \frac{1}{2} \left(2 \cdot \frac{22}{256} \right)^{25} \cong 2^{-64.5} < 2^{-64} .$$

This estimation is for the general MDS_L . The characteristics for MDS_L in Sect. 4.2 is a little better (Sect. 4.2).

$$DP^{XS} \leq 2^{-27.7} , \quad LP^{XS} \leq 2^{-13.8} ,$$

$$DP^{2\text{ rounds}} \leq (2^{-27.7})^5 \cong 2^{-138.3} \ll 2^{-128} ,$$

$$LP^{2\text{ rounds}} \leq \frac{1}{2} (2 \cdot 2^{-13.8})^5 \cong 2^{-65.2} < 2^{-64} .$$

The above result shows that all two-round differential characteristics are far below the critical noise level 2^{-128} , and that the maximum 2-round linear characteristic is equal order of the critical noise level 2^{-64} . Then, we recommend to use Hierocrypt with no less than 6 rounds, where intermediate 4 rounds are for sufficiently small characteristics, and 2 remaining rounds on both ends against partial exhaustive key search.

5.2 Other Cryptanalysis

Besides the differential and the linear cryptanalysis discussed in the previous subsection, there are many other attacks. We discuss the security against the SQUARE (dedicated) attack and the higher-order differential cryptanalysis here [4,8].

The SQUARE attack is presented by J.Daemen, L.R.Knudsen, and V.Rijmen in their paper proposing the block cipher SQUARE [4]. The attack is applicable to SQUARE and other ciphers with similar SPN structures, such as Rijndael or CRYPTON [5,9]. The attack is applicable to Hierocrypt as well, because of its SPN structure. Here, we regard one round of Hierocrypt as two “half-rounds” to compare the strength with SQUARE⁴.

The basic version of SQUARE attack is applicable to 4-round SQUARE and 4-round Rijndael. The attack is based on the following property, that all (16) bytes of the 3rd round output are always balanced over the Λ -set input with only one active byte. Thus, the 128 key bits can be identified by the basic attack with 2^9 plaintexts and 2^9 encryption time for SQUARE and Rijndael.

On the other hand, only two bytes of the 3rd half-round output are always balanced on the same condition. This property reduces to the fact that only 64 key bits can be identified by the basic attack with 2^{11} plaintexts and 2^{11} encryption time for Hierocrypt(Type-II). Thus, the efficiency of basic attack for Hierocrypt(Type-II) is 1/8 of that for SQUARE/Rijndael.

Further study shows that Hierocrypt is stronger in any extended versions of the dedicated attack.

Next, we discuss about the higher differential cryptanalysis. It is known that the cryptanalysis is applicable to \mathcal{KN} cipher, which is provably secure against the differential and linear cryptanalysis[8]. The security against the higher differential cryptanalysis is estimated by the algebraic order. The algebraic order of Hierocrypt’s S-box is 7. And the order after 3 S-box layers (1.5 layers in Hierocrypt convention) is roughly estimated as

$$7^3 = 343 \gg 128 .$$

Therefore, the cryptanalysis does not seem to be feasible.

6 Performance

The cipher Hierocrypt is implemented in four Microprocessors. The measured encryption rates are shown in Table 4.

As the implementation here is rather basic one, the performance is expected to improve by optimizing the implementation.

⁴ One half-round of Hierocrypt corresponds to one round of usual SPN cipher, as it contains one S-box layer

Table 4. Speed of Type-I Hierocrypt encryption(8 rounds)

| Processor | Freq | Platform | Compiler | Throughput |
|----------------|--------|--------------------|------------|------------|
| Pentium III | 550MHz | WindowsNT 4.0 | VC++ 6.0 | 43.18 Mbps |
| Celeron | 466MHz | Linux kernel-2.2.5 | egcs 1.1.2 | 31.74 Mbps |
| Ultra SPARC II | 296MHz | Solaris 2.5.1 | gcc 2.95.1 | 15.90 Mbps |
| Alpha 21164A | 599MHz | Digital UNIX V4.0D | gcc 2.95.1 | 27.04 Mbps |

Table 5. Speed of Type-II Hierocrypt encryption(8 rounds)

| Processor | Freq | Platform | Compiler | Throughput |
|----------------|--------|--------------------|------------|------------|
| Pentium III | 550MHz | WindowsNT 4.0 | VC++ 6.0 | 40.33 Mbps |
| Celeron | 466MHz | Linux kernel-2.2.5 | egcs 1.1.2 | 29.00 Mbps |
| Ultra SPARC II | 296MHz | Solaris 2.5.1 | gcc 2.95.1 | 19.10 Mbps |
| Alpha 21164A | 599MHz | Digital UNIX V4.0D | gcc 2.95.1 | 48.00 Mbps |

7 Concluding Remarks

We propose the block encryption algorithm “Hierocrypt” based on a two-level nested SPN structure. We use MDS mappings for both-level permutations, which assures the minimum number of active S-boxes hierarchically. SQUARE and Rijndael can be regarded as the nested SPN ciphers, where the higher- and lower-level diffusion layers (MDS_H and MDS_L) are made by using the same MDS matrix. On the other hand, the diffusion layers of both levels are designed independently for Hierocrypt. This independency is profitable to improve the security against many attacks including the SQUARE dedicated attack.

Appendix

A Improved Algorithm

We have designed a revised version named “Hierocrypt-3” based on the Type-II algorithm. All components except for the nested SPN structure of the data randomization part are modified. The following modified components of data randomization are presented in this Appendix.

1. S-box
2. Higher-Level Diffusion MDS_H
3. Lower-Level Diffusion MDS_L

A.1 S-Box

The new S-box $S(x)$ is given by the following table in hexadecimal notation.

$$(S(0) \ S(1) \ \cdots \ S(F) \ S(10) \ \cdots \ S(FF))$$

```

= ( 07 FC 55 70 98 8E 84 4E BC 75 CE 18 02 E9 5D 80
    1C 60 78 42 9D 2E F5 E8 C6 7A 2F A4 B2 5F 19 87
    0B 9B 9C D3 C3 77 3D 6F B9 2D 4D F7 8C A7 AC 17
    3C 5A 41 C9 29 EDDE 27 69 30 72 A8 95 3E F9 D8
    21 8B 44 D7 11 0D 48 FD 6A 01 57 E5BD 85 EC 1E
    37 9F B5 9A 7C 09 F1 B1 94 81 82 08 FB C0 51 0F
    61 7F 1A 56 96 13 C1 67 99 03 5E B6 CA FA 9E DF
    D6 83 CCA2 12 23 B7 65 D0 39 7D 3B D5 B0 AF 1F
    06 C8 34 C5 1B 79 4B 66 BF 88 4A C4 EF 58 3F 0A
    2C 73 D1 F8 6B E6 20 B8 22 43 B3 33 E7 F0 71 7E
    52 89 47 63 0E 6D E3 BE 59 64 EE F6 38 5C F4 5B
    49 D4 E0 F3BB 54 26 2B 00 86 90 FF FE A6 7B 05
    AD 68 A1 10 EB C7 E2 F2 46 8A 6C 14 6E CF 35 45
    50 D2 92 74 93 E1 DAAE A9 53 E4 40 CDBA 97 A3
    91 31 25 76 36 32 28 3A 24 4CDBD9 8D DC 62 2A
    EA 15 DDC2 A5 0C 04 1D 8F CB B4 4F 16 ABAA A0) .

```

$$s(x) = \text{Add}(\text{Power}(\text{Perm}(x))) .$$

where Perm is a bit permutation

$$\begin{aligned} \text{Perm} &: \text{GF}(2)^8 \mapsto \text{GF}(2)^8 , \\ y_i &= x_{\pi(i)} , \end{aligned}$$

Table 6. Bit permutation for S-box

| | | | | | | | | |
|----------|---|---|---|---|---|---|---|---|
| i | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| $\pi(i)$ | 3 | 7 | 5 | 8 | 6 | 2 | 4 | 1 |

Power is the power of 247 over $\text{GF}(2^8)$ with the primitive polynomial $z^8 + z^6 + z^5 + z + 1$.

$$\begin{aligned} s &: \text{GF}(2^8) \mapsto \text{GF}(2^8) , \\ s(x) &= x^{247} , \end{aligned}$$

Add is a constant addition.

$$\begin{aligned} \text{Perm} &: \text{GF}(2)^8 \mapsto \text{GF}(2)^8 , \\ \text{Add}(x) &= x \oplus 0x11 . \end{aligned}$$

The power function Power has the same probabilities, both the bit permutation Perm and the constant addition Add do not change them.

Perm is chosen so that the number of polynomials of output bits is the maximum, in order to improve the security against the interpolation attack.

Add is chosen so that the distribution of input and output hamming distances is nearest to that of random function, in order to remove the statistical bias.

The main purpose of S-box modification is to improve the security against the differential and linear cryptanalysis. The maximum differential and linear probabilities of the S-box are 2^{-6} and 2^{-4} , respectively, which are proven to

be minimum theoretically. The maximum differential and linear characteristic probabilities for two rounds are respectively estimated as 2^{-150} and 2^{-76} when the above S-box is used. This estimation indicates that both differential and linear characteristic probabilities saturate after two rounds.

A.2 Lower-Level Diffusion MDS_L

The lower-level diffusion MDS_L is given as follows.

$$MDS_L : GF(2^8)^4 \mapsto GF(2^8)^4 ,$$

$$\begin{pmatrix} y_{i1} \\ y_{i2} \\ y_{i3} \\ y_{i4} \end{pmatrix} = \begin{pmatrix} C4 & 65 & C8 & 8B \\ 8B & C4 & 65 & C8 \\ C8 & 8B & C4 & 65 \\ 65 & C8 & 8B & C4 \end{pmatrix} \begin{pmatrix} x_{i1} \\ x_{i2} \\ x_{i3} \\ x_{i4} \end{pmatrix}$$

We have chosen this maxtrix from circulant MDS matrices so that the output of SP-function (composite function of S-box and MDS_L) has the maximum number of polynomials.

A.3 Higher-Level Diffusion MDS_H

The higher-level diffusion MDS_H is given as follows.

$$MDS_H(X) = D_H X , \quad D_H = D_h \otimes I_8 ,$$

$$D_h = \begin{pmatrix} 5 & 5 & A & E \\ E & 5 & 5 & A \\ A & E & 5 & 5 \\ 5 & A & E & 5 \end{pmatrix} ,$$

$$\begin{pmatrix} y_{11} \\ y_{12} \\ y_{13} \\ y_{14} \\ y_{21} \\ y_{22} \\ y_{23} \\ y_{24} \\ y_{31} \\ y_{32} \\ y_{33} \\ y_{34} \\ y_{41} \\ y_{42} \\ y_{43} \\ y_{44} \end{pmatrix} = \begin{pmatrix} 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 1 & 0 & 1 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 1 & 1 & 1 & 0 & 1 & 0 & 1 \\ 1 & 1 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 1 & 0 & 1 \end{pmatrix} \begin{pmatrix} x_{11} \\ x_{12} \\ x_{13} \\ x_{14} \\ x_{21} \\ x_{22} \\ x_{23} \\ x_{24} \\ x_{31} \\ x_{32} \\ x_{33} \\ x_{34} \\ x_{41} \\ x_{42} \\ x_{43} \\ x_{44} \end{pmatrix} .$$

The MDS matrix is chosen so that any byte is connected to all bytes after one round (one MDS_L and one MDS_H) through more than one intermediate bytes. This condition is imposed to improve the security against the SQUARE dedicated attack. Our evaluation shows that the condition makes the nested SPN cipher at least one half-round stronger than SQUARE and Rijndael against the attack.

References

1. S.Vaudenay, "On the need for multipermutations": Cryptanalysis of MD4 and SAFER. Fast Software Encryption(2), LNCS **1008**, pp.286-297, 1995.
2. V.Rijmen, J.Daemen, B.Preneel, A.Bosselaers, E.DcWin, "The Cipher SHARK," Fast Software Encryption(3), LNCS **1039**, pp.99-112, 1996.
3. A.M.Youssef, S.Mister, and S.E.Tavres, "On the Design of Linear Transformations for Substitution Permutation Encryption Networks," Selected Areas in Cryptography, SAC'97, Workshop Report, <http://saturn.ee.queensu.ca:8000/sac/sac97/papers/paper29.ps>.
4. J.Daemen, L.R.Knudsen, V.Rijmen, "The block cipher Square," Fast Software Encryption(4), LNCS **1267**, pp.149-165, 1997.
5. J.Daemen, V.Rijmen, "AES Proposal: Rijndael," <http://www.esat.kuleuven.ac.be/~rijmen/rijndael/rijndaeldocV2.zip>
6. E.Biham and A.Shamir, "Differential cryptanalysis of DES-like cryptosystems," Journal of Cryptology, **4** No.1, pp.3-72, 1991.
7. M.Matsui, "Linear cryptanalysis method for DES cipher," Eurocrypt'93, LNCS **765**, pp.386-397, 1994.
8. L.R. Knudsen, "Truncated and higher order differentials," Fast Software Encryption(2), LNCS **1008**, pp.196-211, 1995.
9. C.D'Halluin, G. Bijmens, V. Rijmen, B. Preneel, "Attack on 6 rounds of Crypton," Fast Software Encryption(6), LNCS.1636, pp.46-59, 1999.

Symmetric Block Ciphers Based on Group Bases

Valér Čanda¹, Tran van Trung¹, Spyros Magliveras², and Tamás Horváth³

¹ Institute for Experimental Mathematics, University of Essen,
Ellernstrasse 29, 45326 Essen, Germany
{valer, trung}@exp-math.uni-essen.de

² Department of Computer Science and Engineering, University of Nebraska,
Lincoln, NE 68588, USA
spyros@helios.unl.edu

³ Secunet AG, Steubenstrasse 53, 45138 Essen, Germany
horvath@secunet.de

Abstract. We introduce a new family of symmetric block ciphers based on group bases. The main advantage of our approach is its full scalability. It enables us to construct, for instance, a trivial 8-bit Caesar cipher as well as a strong 256-bit cipher with 512-bit key, both from the same specification. We discuss the practical aspects of the design, especially the choice of carrier groups, generation of random group bases and an efficient factorization algorithm. We also describe how the cryptographic properties of the system are optimized, and analyze the influence of parameters on its security. Finally we present some experimental results regarding the speed and security of concrete ciphers from the family.

1 Introduction

A good block cipher should possess several properties. In addition to *security* and *efficiency*, which are essential, there are other important attributes like *generality*, *scalability* and *theoretical foundations*. In what follows we discuss these properties in more detail.

A block cipher can be characterized by two basic parameters: the block length n and the key length k , both expressed as a number of bits. For each of the 2^k possible keys, the cipher defines a bijective mapping between the 2^n plaintext blocks and the 2^n ciphertext blocks. As the plaintext and ciphertext spaces are usually the same, we can view an n -bit block cipher as defining a permutation on a set of 2^n elements for each possible key. A simple key-indexed lookup table containing all 64-bit numbers in random order would implement a very strong 64-bit block cipher, without any additional algorithm. Unfortunately, such an implementation would take so much memory, that it would not be applicable for any practical use. Almost all modern block ciphers simulate such a large lookup random table using smaller tables (S-Boxes) in combination with other transformations. The goal is to make the dependence between the plaintext, ciphertext and key so complex that it is virtually indistinguishable from the random case.

As 2^n elements can be permuted in $2^n!$ different ways, a “perfect” n -bit block cipher could accept keys of length up to $\lfloor \log_2(2^n!) \rfloor$ bits. This means a 1683-bit key for $n = 8$ and approximately a 10^{21} -bit key for $n = 64$. Of course, no one needs such long keys and it would be extremely impractical to use them. These large numbers show, however, the strong potential of block ciphers and the restricted generality of current systems which use by design a fixed key-length or fixed S-boxes. These ciphers are in our opinion not flexible enough. They are constrained to one specific configuration and the functions defined by them might be far from random permutations.

Another frequent drawback of block ciphers is a small or totally missing scalability. Because of the unprecedented growth rate of computer power available to the public, it is highly desirable to have choices for some basic parameters of the cipher. If our ciphers were fully scalable, we could just adapt the values of these parameters, when some new, amazing breakthrough in processor or memory technology occurs. The values of the parameters n and k could be easily changed without a complete redesign of the cipher and we would not be forced to throw away all research on the properties of the cipher, starting again from the beginning.

For example, the block length of DES is 64 bits. If we wish to create a 128-bit version of DES, we would have to design new, larger S-Boxes. As the design of good S-Boxes is by far a non-trivial task [1], the properties of the new DES could be quite different. This, in fact, would be a totally new cipher. Another cipher - IDEA [2] - has a very plain structure and its block length might be doubled simply by increasing the length of each of the four subblocks from 16 to 32 bits. The fact, however, that $2^{16} + 1$ is a prime number is essential to the functionality of IDEA. Since $2^{32} + 1$ is not a prime, the double version would not work well. In contrast to DES and IDEA there are already some nice examples of well-founded scalable designs available like RC5 [12], RC6 [13] or Rijndael [14].

Each new cipher should be studied extensively, perhaps for several years, before it is deemed trustworthy and is presented for widespread use. If the cipher is based on a strong theoretical foundation, we can gain a better understanding of possible failures, cryptanalytic attacks, etc., and we have stronger tools with which to analyze the new algorithm. Therefore, a cipher based on strong mathematical foundations will either be rejected outright, or if it appears workable, there should be a reasonable chance for it to have provable reliability and trustworthiness.

All in all, we think that an ideal cipher should not only be secure and fast but also theoretically well-founded, general, and scalable. In this paper we present a new family of fully scalable block ciphers which is quite general. Our approach enables the construction of a range of ciphers, from a tiny toy cipher to a large, secure one. The idea on which the encryption is based is a mapping of group elements between two random group bases. A subject which does not know the two secret bases is not able to recover the mapping. We discuss the selection of suitable carrier groups, the generation of random group bases which enable an efficient factorization and the optimization of the cryptographic properties of

the system. Finally, we discuss the security, speed and memory requirements of a concrete software implementation.

2 The Principle of Encryption

The ciphers we propose utilize group theory [3]. Although we focus our attention on permutation groups, it is also possible to construct a cryptosystem based on any carrier group in other representation forms. By a *permutation* of n symbols we understand a bijective function $p : \mathbb{Z}_n \rightarrow \mathbb{Z}_n$. The *Cartesian representation* of p is the vector $[p(0), p(1), \dots, p(n-1)]$. This is the usual way to represent permutations in computers. Operation $*$ on permutations is defined as follows: $[p(0), \dots, p(n-1)] * [q(0), \dots, q(n-1)] = [q(p(0)), \dots, q(p(n-1))]$. The basic notion needed for the ciphers proposed in this paper is the idea of a *Group Basis*.

Definition 1. Group Basis

Let G be a finite group. A group basis for G is an ordered collection $\beta = (B_0, B_1, \dots, B_{w-1})$ of ordered subsets $B_i = (b_{i,0}, b_{i,1}, \dots, b_{i,r_i-1})$ of G such that each element $p \in G$ can be expressed uniquely as a product of the form:

$$p = b_{0,x_0} \cdot b_{1,x_1} \cdots b_{w-1,x_{w-1}}, \quad b_{i,x_i} \in B_i$$

The B_i are called the *blocks* of β , the vector of block lengths $r = (r_0, r_1, \dots, r_{w-1})$ is called the *type* of β and the number w the *dimension* of β . Each $p \in G$ corresponds to a unique *index vector* $x = (x_0, x_1, \dots, x_{w-1})$, where $x_i \in \mathbb{Z}_{r_i}$. The space of all index vectors is $X = \mathbb{Z}_{r_0} \times \mathbb{Z}_{r_1} \times \cdots \times \mathbb{Z}_{r_{w-1}}$. The index set X has cardinality $|X| = r_0 \cdot r_1 \cdots r_{w-1} = |G|$.

A basis β describes a bijective mapping $\tilde{\beta} : X \rightarrow G$ as follows:

$$\tilde{\beta}(x) = \tilde{\beta}(x_0, x_1, \dots, x_{w-1}) = b_{0,x_0} \cdot b_{1,x_1} \cdots b_{w-1,x_{w-1}} = p.$$

When computing $p = \tilde{\beta}(x)$ we say that p is *composed* from factors b_{i,x_i} . Computing the inverse function $x = \tilde{\beta}^{-1}(p)$ is called *factorizing* p with respect to β .

It should be noticed that the concept of *group basis* in this paper strongly differs from the notion *base* given in the standard literature of group theory (see e.g. [3]). For more discussion of group bases defined here (also called *logarithmic signatures* in [5]) the reader is referred to [5] and [6].

One can think of a group basis as a kind of w -dimensional discrete coordinate system as illustrated on Fig. 1. The six permutations of G might be seen as points in a 2-dimensional space. Any one of the six points can be expressed as a unique sum¹ of two points, one from each axis. The two axes, the first with three and the second with two points, correspond to the two blocks of β .

¹ Addition of points in this discrete geometry is defined by means of vectors as: $(x_1, y_1) + (x_2, y_2) = (x_1 + x_2 \bmod 3, y_1 + y_2 \bmod 2)$. Note that while this is a commutative operation, operation $*$ in S_3 is not. This is only an illustrative example

Example 1. A Group Basis for $G = S_3$

$$G = \{[0, 1, 2], [0, 2, 1], [1, 0, 2], [1, 2, 0], [2, 0, 1], [2, 1, 0]\}$$

| β | | |
|---------|-------------|-----------|
| B_1 | $[0, 2, 1]$ | $b_{1,1}$ |
| | $[0, 1, 2]$ | $b_{1,0}$ |
| B_0 | $[1, 2, 0]$ | $b_{0,2}$ |
| | $[2, 0, 1]$ | $b_{0,1}$ |
| | $[0, 1, 2]$ | $b_{0,0}$ |

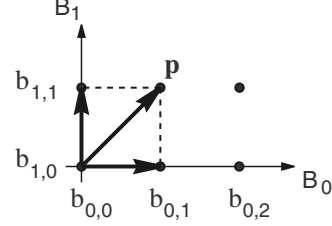


Fig. 1. A coordinate system

$$\begin{aligned} w &= 2, r = (3, 2), \\ p &= [1, 0, 2] = [2, 0, 1] * [0, 2, 1] = b_{0,1} * b_{1,1}, \\ \tilde{\beta}(p) &= (1, 1) = x \end{aligned}$$

The crucial property of group bases from the cryptographic point of view is that *there is an enormous number of different group bases for a given group*. We denote the set of all bases that generate G by \mathcal{B}_G . For example, the tiny group S_3 of our example has 924 different bases. $6!$ of them are one-dimensional bases of type (6) and $2! \cdot 2^3 \cdot 3! + 3! \cdot 3^2 \cdot 2!$ of them are two-dimensional bases of types (2,3) and (3,2). Later we will describe how all these bases can be generated. The most basic version of a secret-key cryptosystem based on group bases is defined as follows:

Definition 2. A Block Cipher Based on Group Bases

Let G be a finite group, called the carrier group. Let $\lambda : \mathbb{Z}_{|G|} \rightarrow G$ be any fixed bijective function. The plaintext and ciphertext spaces for the cipher are the same: $\mathcal{P} = \mathcal{C} = \mathbb{Z}_{|G|}$. The key space is the set $\mathcal{K} = \mathcal{B}_G \times \mathcal{B}_G$.

Let $k = (\beta_1, \beta_2)$, $k \in \mathcal{K}$ be a secret key. Let $x \in \mathcal{P}$ be a plaintext and $y \in \mathcal{C}$ the corresponding ciphertext. The encryption function $e_k : \mathcal{P} \rightarrow \mathcal{C}$ is defined by the rule

$$y = e_k(x) = \lambda^{-1}(\tilde{\beta}_2(\tilde{\beta}_1^{-1}(\lambda(x))))$$

and the decryption function $d_k : \mathcal{C} \rightarrow \mathcal{P}$ is defined as

$$x = d_k(y) = \lambda^{-1}(\tilde{\beta}_1(\tilde{\beta}_2^{-1}(\lambda(y)))).$$

In other words, we take two random group bases for G , β_1 and β_2 , and each time we want to encrypt some $p \in G$, we have to find such $p' \in G$ which has the same coordinates in β_2 as p has in β_1 . The function λ only defines a unique numbering of the group elements.

Again, for a better visualization, we take a simple example with geometric coordinates (Fig. 2). There are 16 numbered points in the space, thus we can encrypt and decrypt the plaintexts and ciphertexts from \mathbb{Z}_{16} . Suppose, we want to encrypt plaintext point 14. First we find the coordinates of the point 14 with respect to basis β_1 . The corresponding index vector is (2,3). Now we compose the point, which has the same coordinates in β_2 , this gives us point 10. Therefore $e_{(\beta_1, \beta_2)}(14) = 10$. The complete table for e_k is displayed on the right hand side.

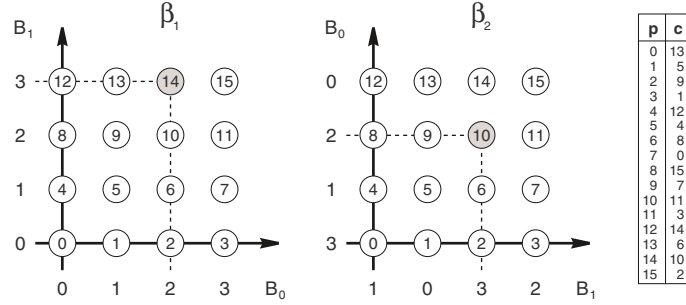


Fig. 2. A mapping of points between two bases

In a real-world application the dependencies become much more complex. A 64-bit cipher can use for instance two 8-dimensional bases with 256 elements on each “axis”. Moreover, the carrier group will not necessarily be commutative.

3 Implementation Aspects

3.1 The Carrier Group G and Function λ

The size of the plaintext and ciphertext space depends directly on the order of the carrier group G . We are only interested in groups whose order is a power of two, the so called 2-groups or binary groups. More precisely, we should have $|G| = 2^{8k}$, for a natural number k , because only ciphers whose blocks fit exactly in k bytes are interesting for a practical use. Note that $|S_m| = m! \neq 2^n$ for any $m > 2$. Therefore the symmetric group S_m is not suitable for a carrier group.

Group \mathbb{Z}_2^n . The simplest available 2-group is the *elementary abelian* group

$$\mathbb{Z}_2^n = \overbrace{\mathbb{Z}_2 \times \mathbb{Z}_2 \times \cdots \mathbb{Z}_2}^{n\text{-times}}.$$

It contains the permutations of $2n$ symbols in form $p = [a_0, a_1, \dots, a_{2n-1}]$ where for each pair of symbols a_{2k}, a_{2k+1} , $k \in \mathbb{Z}_n$ either $a_{2k} = 2k$ and $a_{2k+1} = 2k+1$, or else $a_{2k} = 2k+1$ and $a_{2k+1} = 2k$.

The permutations of \mathbb{Z}_2^n can be represented very efficiently with our so called *compact representation*. The compact representation of a $p = [a_0, a_1, \dots, a_{2n-1}]$, $p \in \mathbb{Z}_2^n$ is the binary vector $x = (x_0, x_1, \dots, x_{n-1})$ where $x_i = 0$ if and only if $a_{2i} = 2i$. Otherwise $x_i = 1$. In other words, the i -th bit of the compact representation indicates, whether the elements a_{2i} and a_{2i+1} have been swapped or not. In terms of memory requirements the compact representation is optimal, as it is impossible to represent the 2^n elements in less than n bits. Another benefit of the compact representation is that it makes it possible to multiply permutations very fast. Note that if x_1 is the compact representation of p_1

and x_2 of p_2 , then $x_1 \text{ XOR } x_2$ is the compact representation of the product $p_1 * p_2$. The operation $*$ in \mathbb{Z}_2^n is commutative and linear. Last but not least, the compact representation of the permutations fulfills the role of the function λ from Definition 2. If we consider the vector $(x_0, x_1, \dots, x_{n-1})$ as a binary representation of a natural number, we have a unique numbering of all permutations in the group.

The group basis for \mathbb{Z}_2^n of the form $\alpha = (A_0, \dots, A_{n-1})$, where each block A_i contains two permutations in compact representation, the identity $(\underbrace{00 \dots 0}_{n \text{ times}})$ and a single swap on the i -th place $(\underbrace{0 \dots 0}_{i-1 \text{ times}} \ 1 \ \underbrace{0 \dots 0}_{n-i \text{ times}})$, is called the *canonical basis* for \mathbb{Z}_2^n . The one-element set $c_i = \{i\}$ is called the set of *key bit positions* for block A_i .

Group $\mathcal{H}_s \times \mathcal{H}_1$. In contrast with \mathbb{Z}_2^n , the most complex 2-group is \mathcal{H}_s , the largest binary subgroup of S_n . When $n = 2^s$, the order of \mathcal{H}_s is 2^{2^s-1} . \mathcal{H}_s is also known as the Sylow 2-subgroup of S_{2^s} .

Definition 3. Sylow 2-subgroup \mathcal{H}_s of the symmetric group S_n , $n = 2^s$.

The group \mathcal{H}_s is defined recursively as follows:

- $\mathcal{H}_1 = \mathbb{Z}_2$
- $\mathcal{H}_s = (\mathcal{H}_{s-1} \times \mathcal{H}_{s-1}) \cdot \mathbb{Z}_2$, for $s > 1$.

The permutation representation \mathcal{T}_s of the \mathbb{Z}_2 appearing in \mathcal{H}_s , contains two permutations of 2^s elements, the identity ι and the involution τ_s , which swaps the two halves $\{0, 1, \dots, 2^{s-1} - 1\}$ with $\{2^{s-1}, \dots, 2^s - 1\}$, each of length 2^{s-1} . For example $\mathcal{T}_1 = \{[0, 1], [1, 0]\}$, $\mathcal{T}_2 = \{[0, 1, 2, 3], [2, 3, 0, 1]\}$, etc.

Example 2. \mathcal{H}_s and α_s for $s = 1, 2, 3$.

$$\begin{aligned} \mathcal{H}_1 &= \mathcal{T}_1 = \{[0, 1], [1, 0]\} & |\mathcal{H}_1| &= 2^{2^1-1} = 2 \\ \mathcal{H}_2 &= (\mathcal{H}_1 \times \mathcal{H}_1) \cdot \mathcal{T}_2 = \{[0, 1, 2, 3], [1, 0, 2, 3], [0, 1, 3, 2], [1, 0, 3, 2], \\ &\quad [2, 3, 0, 1], [2, 3, 1, 0], [3, 2, 0, 1], [3, 2, 1, 0]\} & |\mathcal{H}_2| &= 2^{2^2-1} = 8 \\ \mathcal{H}_3 &= (\mathcal{H}_2 \times \mathcal{H}_2) \cdot \mathcal{T}_3 = \{[0, 1, 2, 3, 4, 5, 6, 7], \dots, [7, 6, 5, 4, 3, 2, 1, 0]\} & |\mathcal{H}_3| &= 2^{2^3-1} = 128 \end{aligned}$$

Each \mathcal{H}_s has a unique *canonical basis* α_s which contains $2^s - 1$ blocks each consisting of two permutations. Each block A_i has one key bit position $c_i = \{i\}$. An α_s is constructed recursively as follows:

$$\begin{aligned} \alpha_1 : & \begin{array}{|c|c|} \hline A_0 & \begin{array}{c} 1 \ 0 \\ 0 \ 1 \end{array} \\ \hline \end{array} & \alpha_2 : & \begin{array}{|c|c|} \hline A_2 & \begin{array}{c} 2 \ 3 \ 0 \ 1 \\ 0 \ 1 \ 2 \ 3 \end{array} \\ \hline A_1 & \begin{array}{c} 0 \ 1 \ 3 \ 2 \\ 0 \ 1 \ 2 \ 3 \end{array} \\ \hline A_0 & \begin{array}{c} 1 \ 0 \ 2 \ 3 \\ 0 \ 1 \ 2 \ 3 \end{array} \\ \hline \end{array} & \alpha_3 : & \begin{array}{|c|c|} \hline A_6 & \begin{array}{c} 4 \ 5 \ 6 \ 7 \ 0 \ 1 \ 2 \ 3 \\ 0 \ 1 \ 2 \ 3 \ 4 \ 5 \ 6 \ 7 \end{array} \\ \hline A_5 & \begin{array}{c} 0 \ 1 \ 2 \ 3 \ 6 \ 7 \ 4 \ 5 \\ 0 \ 1 \ 2 \ 3 \ 4 \ 5 \ 6 \ 7 \end{array} \\ \hline A_4 & \begin{array}{c} 0 \ 1 \ 2 \ 3 \ 4 \ 5 \ 7 \ 6 \\ 0 \ 1 \ 2 \ 3 \ 4 \ 5 \ 6 \ 7 \end{array} \\ \hline A_3 & \begin{array}{c} 0 \ 1 \ 2 \ 3 \ 5 \ 4 \ 6 \ 7 \\ 0 \ 1 \ 2 \ 3 \ 4 \ 5 \ 6 \ 7 \end{array} \\ \hline A_2 & \begin{array}{c} 2 \ 3 \ 0 \ 1 \ 4 \ 5 \ 6 \ 7 \\ 0 \ 1 \ 2 \ 3 \ 4 \ 5 \ 6 \ 7 \end{array} \\ \hline A_1 & \begin{array}{c} 0 \ 1 \ 3 \ 2 \ 4 \ 5 \ 6 \ 7 \\ 0 \ 1 \ 2 \ 3 \ 4 \ 5 \ 6 \ 7 \end{array} \\ \hline A_0 & \begin{array}{c} 1 \ 0 \ 2 \ 3 \ 4 \ 5 \ 6 \ 7 \\ 0 \ 1 \ 2 \ 3 \ 4 \ 5 \ 6 \ 7 \end{array} \\ \hline \end{array} & \alpha_s : & \begin{array}{|c|c|} \hline \hat{I}_{s-1} & I_{s-1} \\ \hline I_{s-1} & \hat{I}_{s-1} \\ \hline J_{s-1} & \hat{\alpha}_{s-1} \\ \hline \alpha_{s-1} & \hat{J}_{s-1} \\ \hline \end{array} \end{aligned}$$

Here, I_s is defined as $(0, 1, \dots, 2^s - 1)$, $\hat{I}_s = 2^s + I_s = (2^s, 2^s + 1, \dots, 2^{s+1} - 1)$ and J_s denotes a $(2^{s+1} - 2) \times 2^s$ array each row of which is equal to I_s . Analogously, $\hat{J}_s = 2^s + J_s$.

Definition 4. The compact representation of elements in \mathcal{H}_s .

Let h be a permutation from \mathcal{H}_s . The binary vector $\tilde{\alpha}_s^{-1}(h) = (x_0, x_1, \dots, x_{w-1})$, $w = 2^s - 1$, is called the compact representation of h .

Again, the compact representation is optimal in term of memory requirements. In general, we can say that each $h \in \mathcal{H}_s$ can be uniquely represented by a $(2^s - 1)$ -bit binary number. The multiplication of permutations in \mathcal{H}_s is a non-linear and non-commutative operation. It can be performed directly and efficiently in the compact representation [6].

As already mentioned, the preferred order of the carrier group should be a number in form 2^{8k} , where $k \in \mathbb{N}$. However, the order of \mathcal{H}_s is 2^{2^s-1} and $2^s - 1 \neq 8k$. Thus the real-world ciphers will be based on a group whose compact representation is one bit longer. This can simply be achieved by using a slightly modified group $\mathcal{H}_s \times \mathcal{H}_1$ instead of \mathcal{H}_s . The compact representation grows by one bit to the desired 2^s and the multiplication stays in principle the same as in \mathcal{H}_s , only the highest bit must be handled (XOR-ed) separately. The multiplication operation continues to be non-commutative and non-linear. From now on we suppose that all permutations and all group bases are stored and manipulated only in the compact representation.

We have presented the two most extreme examples for permutation 2-groups, the simplest \mathbb{Z}_2^n , which is commutative, and the most complex $\mathcal{H}_s \times \mathcal{H}_1$. In principle any other 2-group can be used in an appropriate representation. New 2-groups for our cryptographic purposes can be constructed from the available ones by taking *wreath products*, *direct products*, *extensions* and their combinations [6].

3.2 Key Generation

A key for our cryptosystem consists of two randomly chosen group bases. This approach ensures an extremely high upper limit of the scalable key space. Because the bases can hardly be entered manually by the user, we need a mechanism for generating random bases. Possibly, in cases where a fixed key length is expected, the bases could be generated from a binary key of fixed length or from a pass-phrase, in conjunction with the use of a pseudo-random number generator, which in turn is based on a subsystem implementing a fixed version of our system.

In general, not every basis enables a fast factorization. An efficient factorization algorithm is only known for so called *transversal* group bases [5], [6]. Therefore we want to generate only bases of this kind. The *Basis Generation Algorithm* (BGA) starts from the canonical basis α and carries out the following four steps:

1. The *commutative block shuffle* operation randomly changes the order of the blocks by multiple swaps of two adjacent blocks. Two blocks $B_i = (b_{i,0}, \dots, b_{i,r_i})$ and $B_{i+1} = (b_{i+1,0}, \dots, b_{i+1,r_{i+1}})$ can be swapped only if $b_{i,j} * b_{i+1,k} = b_{i+1,k} * b_{i,j}$ for $j \in \mathbb{Z}_{r_i}$ and $k \in \mathbb{Z}_{r_{i+1}}$.
2. The *block fusion* operation replaces two randomly chosen, adjacent blocks $B_i = (b_{i,0}, \dots, b_{i,r_i-1})$ having the set of key bit positions c_i and $B_j = (b_{j,0}, \dots, b_{j,r_j-1})$, $j = i + 1$, having the key bit positions c_j by a single longer block $B'_i = B_i \times B_j = (b_{i,m} * b_{j,n} : m \in \mathbb{Z}_{r_i}, n \in \mathbb{Z}_{r_j})$ having the key bit positions $c'_i = c_i \cup c_j$. Note that block fusion changes the type of the basis from $r = (r_0, r_1, \dots, r_i, r_{i+1}, r_{i+2}, \dots, r_{w-1})$ to $r' = (r_0, r_1, \dots, r_i \cdot r_{i+1}, r_{i+2}, \dots, r_{w-1})$ and decreases the dimension of the basis from w to $w - 1$.
3. The *randomization* operation replaces each $b_{i,j} \in B_i$, $i \in \{1, 2, \dots, w - 1\}$, $j \in \mathbb{Z}_{r_i}$ by $b'_{i,j} = b_{i,j} * \prod_{k=0}^{i-1} b_{k,l_k}$, where $l_k \in \mathbb{Z}_{r_k}$ is chosen randomly for every combination of i, j and k .
4. The *element shuffle* operation randomly changes the order of the elements within each block.

Each step can be skipped or carried out several times. If $\beta \in \mathcal{B}_G$ then each β' generated from the β by any combination of these steps is also in \mathcal{B}_G . Moreover, BGA preserves transversality, so all bases generated from the transversal α will enable a fast factorization. For instance, the basis β_2 in Fig. 2 was created from β_1 by a block shuffle (the axes B_0 and B_1 are exchanged) and an element shuffle (the indices of the points on each axis are shuffled). Block fusion and randomization were not applied there.

The complete key generation scheme from the pass-phrase to the pair of group bases might look as shown in Fig. 3.

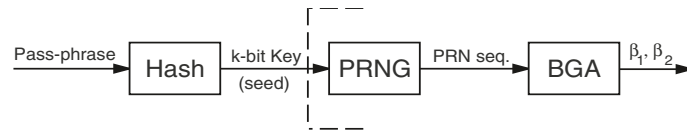


Fig. 3. Key generation

A k -bit hash value is extracted from the pass-phrase which was entered by the user. For example a Cyclic Redundancy Code with a primitive polynomial of degree $k + 1$ might be used for obtaining the k -bit hash. Optionally, the k -bit binary key K can be generated or entered directly. The length of K is freely scalable, theoretically up to several tens of thousands of bits. In practice, lengths of about 64 to 256 bits will be used. Key K is passed as a seed to a pseudo-random number generator which delivers pseudo-random numbers to the BGA. The generator PRNG is a sensitive part of an implementation and must be chosen very carefully. (See also Sect. 4.1.)

3.3 Fast Factorization

Suppose G is a 2-group, $|G| = 2^n$, and $\beta = (B_0, \dots, B_{w-1})$ a transversal, w -dimensional basis of G of type $r = (r_0, \dots, r_{w-1})$. Each block $B_i = (b_{i,0}, \dots, b_{i,r_i-1})$ contains r_i permutations, where $r_i = 2^{m_i}$. The set $c_i = \{c_{i,1}, \dots, c_{i,m_i}\}$, contains the key bit positions for B_i . Let $KB_i : \mathbf{2}^n \rightarrow \mathbf{2}^{m_i}$ be a function which extracts the key bits from a binary vector, $KB_i(a_0, \dots, a_{n-1}) = (a_{c_{i,1}}, \dots, a_{c_{i,m_i}})$.

The factorization of a permutation $p \in G$ is performed level-wise. First, the highest coordinate x_{w-1} is obtained from $p_w = p$ as described below, then the intermediate result $p_{w-1} = p_w * b_{w-1,x_{w-1}}^{-1}$ is passed to the lower level and the process continues in the same way until the lowest level, where an x_0 is obtained and $p_0 = p_1 * b_{0,x_0}^{-1}$ is equal to the identity permutation in G .

Let $p_i = (a_0, \dots, a_{n-1}) \in \mathbf{2}^n$ be an input to a factorization step. The index x_{i-1} is obtained as $x_{i-1} = F_{i-1}(KB_{i-1}(p_i))$, where $F_i : \mathbf{2}^{m_i} \rightarrow \mathbf{2}^{m_i}$ is a bijection such that $F_i(k) = j$ if and only if $KB_i(b_{i,j}) = k$.

One should remark that although there is a similarity between the factorization with respect to a transversal group basis and the Schreier-Sims algorithm working on strong generating sets, they are not equivalent. The concept of group basis is more general than the strong generating set. For comparison see the works [4] and [5].

3.4 Extensions of the Basic System

The cryptosystem introduced in Definition 2 demonstrates the basic principle of encryption based on group bases, the mapping of elements from one basis to another. However, even if we use a non-commutative carrier group with multi-dimensional bases, the cryptographic properties of this mapping will not be sufficient. In the following we present two effective techniques which extend the basic setup and improve the *confusion* as well as the *diffusion* [7] of the cipher.

Bit Reversing. During encryption a permutation $p \in G$ is factorized with respect to the first basis β and the resulting coordinates x_i are passed to the composition in the second basis β' . Let us suppose for a moment, that both β and β' are of the same type, this makes the problem more obvious.

Because both bases are randomized, we can consider each factorization and composition level as a kind of an S-Box, which increases the confusion. As shown on the left side of Fig. 4, each of the indices x_i has been influenced by a different number of S-Boxes. While x_0 passed all eight, x_3 went only through two S-Boxes. That means that some parts of the information contained in p have been “scrambled” much less than other ones. This is an undesirable property, because “parts of the information” are not fully protected. Moreover, if G is a commutative group (such as \mathbb{Z}_2^n), a large part of the ciphertext will not depend on x_3 at all. So diffusion is also reduced.

For this reason, we propose a bit reversing of the index vector x before the start of composition. Note that bit reversing is better than a simple index vector

component reversing, because the bases are not necessarily of the same type. The new encryption and decryption functions are:

$$y = e_k(x) = \lambda^{-1}(\tilde{\beta}_2(R(\tilde{\beta}_1^{-1}(\lambda(x))))))$$

$$x = d_k(y) = \lambda^{-1}(\tilde{\beta}_1(R(\tilde{\beta}_2^{-1}(\lambda(y))))))$$

where the function $R : \mathbf{2}^n \rightarrow \mathbf{2}^n$ reverses the order of the bits of a binary vector. $R(b_0, b_1, \dots, b_{n-1}) = (b_{n-1}, b_{n-2}, \dots, b_0)$, $b_i \in \{0, 1\}$. The effect of R is illustrated on the right hand side of Fig. 4.

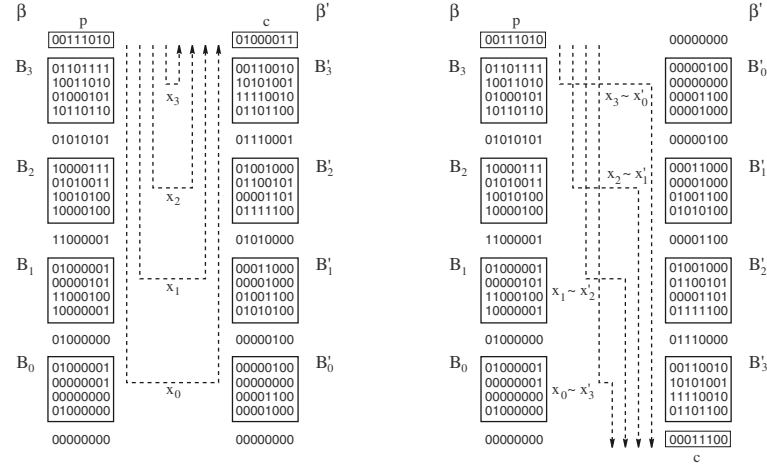


Fig. 4. Effect of bit reversing

Now every index passes exactly five S-Boxes resulting in a balanced confusion of all components. The length of the index vectors $x = x_0 || x_1 || \dots || x_{w_1-1}$ and $y = y_0 || y_1 || \dots || y_{w_2-1}$ is the same $|x| = |y| = n$, even if the bases β and β' are not of the same type. So bit reversing can be used in this case as well.

Non-linear Diffusive Transformation. At each factorization level the input p_i is divided by a factor $b_{i-1, x_{i-1}}$ from the current basis block. Only a small part, namely the key bits, of p_i determines which factor will be taken. Because multiplication and division of permutations in the compact form of \mathbb{Z}_2^n are defined as a simple bit-wise XOR, a change of a single non-key bit of p_i affects only a single bit of p_{i-1} . Consequently diffusion at each factorization step is weak. Even worse, factorization in \mathbb{Z}_2^n is a linear function. Although factorization in \mathcal{H}_s is not linear and its diffusion is the best among all 2-groups, it is still not strong enough from the cryptographic point of view. This is because the higher order bits of a product depend only on the higher order bits of multiplicands.

Fortunately, both the weak diffusion and linearity can be compensated by a simple extension of group bases. Figure 5 shows the idea on a geometric analogy

to the group bases of \mathbb{Z}_2^4 . Basis A was obtained from the canonical α_4 by two fusing B_0 with B_1 and B_2 with B_3 . An element shuffle of A resulted in basis B and applying a further randomization created C . In all three cases the factor $B_1[x_1]$ of any point depends only on its vertical position, the horizontal position does not play any role. The points having the same factors in B_1 lie on parallel horizontal lines (surfaces).

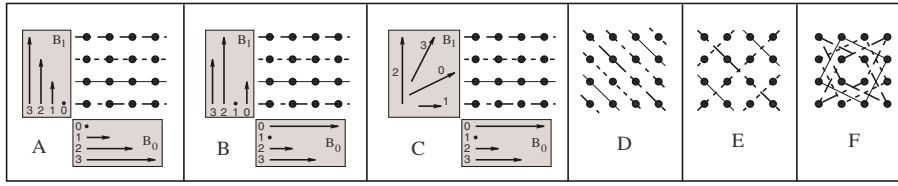


Fig. 5. Linear and non-linear bases

However, one can also construct other, more complex bases. For instance, in basis D , the factor $B_1[x_1]$ depends on both the vertical and the horizontal position of a point. The lines, connecting the points having the same factor in B_1 , are no longer horizontal. Moreover, in bases E and F the surfaces are not even linear. This can be seen as a generalization from an orthogonal two-dimensional coordinate system to a more general geometric coordinatization. (e.g. radial etc.)

Translated back to group bases, before each factorization step the key bits of an intermediate result p_i (as defined in Sect. 3.3) will be made dependent on all bits of p_i . A non-linear hash function $T_i : \mathbf{2}^n \rightarrow \mathbf{2}^{m_i}$ from n to m_i bits will be used for that purpose. Let $c_i = \{c_{i,1}, \dots, c_{i,m_i}\}$ be a set of key bit positions for block B_i and let $SB_i : \mathbf{2}^n \times \mathbf{2}^{m_i} \rightarrow \mathbf{2}^n$ be a function which sets the key bits of a binary vector to a specified value, $SB_i((a_0, \dots, a_{n-1}), (d_1, \dots, d_{m_i})) = (e_0, \dots, e_{n-1})$, such that $e_j = a_j$ for all $j \notin c_i$ and $e_{c_{i,k}} = d_k$ for all $k = 1, \dots, m_i$. A preprocessing step will be then defined as follows: $p'_i = SB_{i-1}(p_i, T(p_i))$. The definition of the factorization step is the same as in Sect. 3.3, with the exception that the transformed value p'_i is processed instead of the original p_i .

In the cases A, B, C of our example, $KB_1 : \mathbf{2}^4 \rightarrow \mathbf{2}^2$ extracted two key bits from a 4-bit word (compact representation of a $p \in \mathbb{Z}_2^4$), $KB_1(a_0, a_1, a_2, a_3) = (a_2, a_3)$, and $F_1 : \mathbf{2}^2 \rightarrow \mathbf{2}^2$ found the appropriate factor in B_1 , $F_1(a_2, a_3) = x_1$. In the extended version (cases D, E, F) x_1 depends on all four bits of p . In case C , $T_1 : \mathbf{2}^4 \rightarrow \mathbf{2}^2$ is defined by $T_1(a_0, a_1, a_2, a_3) = (a_0, a_1) + (a_2, a_3)$, in case D by $T_1(a_0, a_1, a_2, a_3) = (a_0, a_1) \text{ XOR } (a_2, a_3)$ and in case E as $T_1(a_0, a_1, a_2, a_3) = \text{ROTL}(a_0, a_1) + (a_2, a_3)$. Many other functions are also possible. In general, the hash function T should possess at least the following four properties:

- each bit of the output should be dependent on each bit of the input
- each output bit should be balanced²

² An output bit is balanced if $n_0 \cong n_1$, where n_a is the number of inputs for which the output bit is equal to a

- the function should not be linear
- composite function $SB(p, T(p))$ must be invertible

However, one can also use stricter criteria, similar to those used in the construction of an $n \times m_i$ S-Box [1]. When using a proper T , the avalanche effect in the cipher will be very strong, because every single factorization or composition step ensures that the avalanche criterion is fulfilled. In [11] the author defines the term *excess avalanche factor* for iterative ciphers. If we think of every factorization or composition step of our non-iterative ciphers as a “round”, then the value of an analogue of the EAF will be equal to $w_1 + w_2$, where w_i are the dimensions of used bases.

4 Security Aspects

The security, speed and memory requirements of our ciphers depend strongly on the concrete configuration. The most important parameters are:

- the order of the carrier group (affects the block length),
- the extent of block fusion (affects the size of “S-Boxes” and the number of “rounds”),
- the function T used,
- and the randomness of the group bases.

By implementations, where i) the carrier group has large order (e.g. 2^{128}), ii) the extent of block fusion is reasonably large (e.g. 12 key bits per block), iii) a sensible non-linear T was chosen and iv) the bases were generated directly from some physical source of “true” random numbers, a high degree of security is ensured.

4.1 The Pseudo-Random Number Generator

A key in our cryptosystem consists of two secret group bases, alternatively speaking, of two sets of several, large, key dependent S-Boxes of special structure. As the algorithm itself is simple and public, a possible attack would try to reconstruct the bases, using a chosen plaintext attack or similar techniques.

If an implementation uses a PRNG for generating the bases, the properties of the PRNG are crucial for the security of the cipher. The number of possible initial states of the generator, i.e. the size of the generator’s seed, must be reasonably high, because it directly bounds the real key space of the cryptosystem, which must be exhausted in a brute force attack. Of course, we cannot use a simple 32-bit linear congruential generator, unless we want to construct a weak cipher. The size of the generator’s seed is just one of the many measures that determine the quality of the PRNG from a cryptographic point of view. The PRNG needs to pass non-trivial randomness tests, like the Maurer Test [8], the Diehard suite [9], etc. Otherwise some attacks based on dependencies within the bases might be possible.

In our opinion, the lagged Fibonacci generator with Lüscher's approach [10] is a proper example of an acceptable PRNG. For instance, using lags (37,100) with a word length of 32 bits, the generator passes all statistical tests, the size of its seed can be scaled up to 3200 bits and the period of the generated sequence is 2^{131} . These values can be further improved by changing the lags.

Finally, a comment should be made about a brute force attack on a cipher using BGA. The time needed for generating bases (usually less than one second) is negligible for the legal user, who generates the key once, but it is a big problem for an attacker, who tries all possible keys. When trying, say, 2^{64} different keys, with a delay of 500 ms per key, a brute-force attack is infeasible.

4.2 Block Fusion

The average length of blocks is also very important for the security of the system. Let 2^n be the order of the group G and let x be a divisor of n . When BGA merges x adjacent blocks $k \cdot x, k \cdot x + 1, \dots, k \cdot x + x - 1$, of $\alpha_n k \in [0, \frac{n}{x} - 1]$, we say that a block fusion to extent x was performed. (x is equal to the number of key bits per block.) The number of adjacent blocks merged needs not necessarily be constant for all fused x -tuples. In this case the average fusion extent can be computed by $x = \frac{n}{w}$, where w is the dimension of the basis after block fusion.

For instance the canonical basis α_{64} has 64 blocks with two permutations in each block. Each permutation in the compact form is 64 bits long, so the whole basis fits in 1 KB of memory. If we perform a fusion to extent 4, we obtain a basis with 16 blocks of 16 permutations (2 KB), a fusion to extent 8 creates 8 blocks of 256 permutations (16 KB), etc. The fusion to extent 64, would result in one block of 2^{64} permutations (2^{27} TB). Of course, the higher the extent of block fusion, the more secure the cipher. In the extreme case $x = n$ we obtain a full random permutation of 2^n elements, which is the strongest n -bit cipher available. On the other hand, the memory requirements are growing exponentially with x and the quality of the PRNG becomes more critical. The more PRNs are generated, the higher the probability, that some weakness of the PRNG might be exploited. For the reasons above a tradeoff between security and memory load must be found. The values between 8 and 16 key bits per block would be appropriate for practical use.

4.3 Simplified Variants

For speed optimization one could use some simplified variants from our general family of block ciphers, e.g. by taking simple T , using fixed key bits positions or fixed base-block size. However, one should be careful about it, because some of these simplifications might compromise the security. For example, it is not clear whether it is secure to use a BGA without block shuffling in combination with the \mathbb{Z}_2^n carrier group. The fixed key bits positions enable faster implementation, but given a concrete transformation T it might be possible to construct a differential which passes $w - 1$ factorization steps. Consequently a differential attack might be possible. However, to be able to break the cipher an attacker would have to

completely reconstruct both secret group bases which are several kilobytes long. This does not seem to be a practical attack even for a simplified cipher.

5 Experimental Results

We implemented a scalable software version of the proposed algorithm. We used two carrier groups, the $\mathcal{H}_s \times \mathcal{H}_1$, supporting block lengths 8, 16, 32, 64, 128 and 256 bits, as well as the \mathbb{Z}_2^n , supporting all block lengths from 32 to 512 bits, divisible by 32. We used a fixed key length of 128 bits given by the number of possible initial states of the PRNG.

5.1 Throughput

Both algorithms were implemented in C++ and tested on a Pentium II machine running at 350 MHz. As expected, the first carrier group was less suitable for a software implementation. The multiplication of permutations from $\mathcal{H}_s \times \mathcal{H}_1$ in the compact representation is a bit-oriented recursive algorithm not very well supported by the instruction set of the processors. To make the factorization faster, we precomputed the inverses of all permutations in the group bases. So we actually stored four instead of two bases. The required memory space was about 90 KB. The throughput of the 64-bit version without transformation T was about 75 KB/s and with a simple transformation only 50 KB/s. When we used the Cartesian representation of permutations, speeds rose to 275 KB/s without a T , and 100 KB/s with a transformation. The memory requirements were about 900 KB. Even if some tighter optimization techniques were to improve the speeds by a factor of 2 to 4, the values achieved by the software implementation can not be considered as very satisfactory. A simplified hardware version of the algorithm with its own special multipliers running at a clock rate of 45 MHz achieves speeds above 20 MB/s according to [6], so the group $\mathcal{H}_s \times \mathcal{H}_1$ is definitely more suitable for a hardware implementation.

Our second implementation used \mathbb{Z}_2^n as carrier group. The multiplication of permutations in this commutative group is much faster than in $\mathcal{H}_s \times \mathcal{H}_1$. We used a simplified BGA without block shuffle and with the fixed fusion length 8 blocks, which made the factorization even more efficient. A non-linear transformation T was used at each level of the factorization and composition operations. The 64-bit version occupied 18 KB and encrypted at a rate of 2 MB per second. The 128-bit version with memory requirements 69 KB achieved about 1.5 MB/s and the 256-bit version ran at 1 MB/s occupying 270 KB of memory. Again some speed improvements by a factor of 2 to 3 might be possible after a strong optimization effort. These results confirmed that \mathbb{Z}_2^n is much more efficient than $\mathcal{H}_s \times \mathcal{H}_1$, at least in software.

Unfortunately, the encryption speeds measured by \mathbb{Z}_2^n are approximately 2 to 7 times slower than the speeds of recent fast block ciphers. Even if we consider some minor possible optimizations, the achieved speeds are not satisfactory.

5.2 Randomness

We used a general statistical approach to estimate the quality of encryption. Similar methods have already been used in several works, for example in [15]. Of course, this approach can not replace a deep analysis of the cipher, but it at least gives us a good estimate of cipher's quality. In our tests we encrypted a large amount of highly redundant non-periodic data (e.g. a sequence of blocks, containing n -bit binary representation of a counter sequence 0, 1, 2, ...), and tested the output for randomness. The idea of the testing approach is the following: The cipher must provide strong diffusion, so even small changes between the adjacent input blocks must result in big and random looking changes in the output blocks. Further, the cipher must provide a strong confusion, so a systematic and highly redundant input sequence must be encrypted into a sequence which can not be distinguished from a true random one by any statistical tests. The output sequences were tested by the DieHard suite of statistical tests [9]. The tests were carried out for many different keys.

The data were encrypted using the carrier group \mathbb{Z}_2^n and the bases were generated with the lagged (37, 100) Fibonacci Generator with Lütcher's approach. The fixed number of key bits per block was set to 8. Here is a C-like definition of one of the non-linear transformations used, $T : \mathbf{2}^n \rightarrow \mathbf{2}^8$, $n = 8k$, $k \in \mathbb{N}$:

```
byte T(vector p, int n)
    byte sum = 0;
    for i = 0 to n / 8 - 1
        sum = rotr3(sum + p[i]);
    return sum;
```

The $p[i]$ are the 8-bit segments (bytes) of the n -bit binary vector p and the function `rotr3` performs a 3-bit right rotation of an 8-bit value.

Each test from the DieHard suite evaluates the quality of the input with a so called p -value, $p \in [0, 1]$. Good results should lie between 0.001 and .999. The tests with results below 10^{-6} or above $1 - 10^{-6}$ are considered as failed. However, one must keep in mind that even a true random number generator generates sometimes a sequence, which “fails” the test, since all sequences, even the “less random” ones, appear with the same probability.

We carried out 4400 tests for each configuration and counted the number of significant results among these tests. A result was considered as significant (or suspect), if the value p was below 0.001 or above 0.999. The average ratio of significant results, measured by our cipher, was 0.0024 for block length 64 bits and 0.0025 for 128 bits. In our opinion the results can be considered as satisfactory. For instance, the cipher IDEA, which is regarded as one of the most secure 64-bit ciphers today, achieved on average 0.0029 of significant results by the same test. An output of a simple linear congruential generator produced 0.5036 of significant results.

6 Conclusions

We have introduced a new framework for constructing block ciphers based on group bases. Our approach enables to design a simple weak cipher, which can be deeply analyzed and examined, as well as a large, strong one. Even the full symmetric group of degree 2^n can be realized from the same specification.

In contrast to Feistel networks, our ciphers are not iterative. Instead of several repetitions of a uniform round, a specific number of factorization and composition steps are carried out. The security can be scaled through the average fusion extent instead of the number of rounds. As the key of the cipher is in fact a random pair of group bases, the potential size of the key space is much larger than by the “classical” ciphers. In practice the group bases will be generated from some fixed-size seed value, so they can be viewed as a set of random key-dependent S-Boxes with a special structure.

The block length, key length and security level of the ciphers are scalable. Some other components of the cipher, which affect the speed and memory requirements, are also variable. The system has been optimized for maximal confusion, diffusion and non-linearity. The results of statistical tests were very satisfactory. Nevertheless, the cryptosystem is still too new to allow us to make strong statements about its security. Some attacks based on the special structure of the group bases may be possible as well as attacks targeting the special properties of used PRNG. The presence of a mathematical foundation lets us hope that a deeper theoretical analysis of the cryptosystem will be possible.

There are still some open questions about the new design, for instance:

- Compared to the fast modern block ciphers the proposed ciphers are rather slow. Is it possible to significantly improve their speed?
- Although the general design appears to be very robust, it is not clear whether it is also true for the simplified variants (e.g. using fixed key bits positions, fixed base block length, etc.). How is their resistance to the differential and the linear cryptanalysis?
- What is the minimal block fusion extent, which provides a strong security?

Research on these problems is likely to be the subject of some future work on this area.

References

1. C. Adams, S. Tavares, Structured design of cryptographically good S-Boxes, in *Journal of Cryptology*, **3** (1990).
2. X. Lai, On the Design and security of block ciphers, in *ETH Series in Information Processing*, **1** (1992).
3. J. D. Dixon, B. Mortimer, Permutation groups, *Springer Verlag*, (1996).
4. C. C. Sims, Computation with permutation groups, in *Proc. Second Sympos. on Symbolic and Algebraic Manipulation*, Assoc. Comput. Mach., (1971).
5. S. S. Magliveras, N. D. Memon, The algebraic properties of cryptosystem PGM, in *Journal of Cryptology*, **5** (1992), pp 167-183.

6. T. Horváth, Cryptosystem TST - Ph.D. thesis, in *University of Essen, Germany*, (1998), <http://www.exp-math.uni-essen.de/~trung/tst/>.
7. C. E. Shannon, Communication theory of secrecy systems, in *Bell System Technical Journal*, **28** (1949), pp 656–715.
8. U. M. Maurer, A universal statistical test for random bit generators, in *Journal of Cryptology*, (1992), pp 89–105.
9. G. Marsaglia, Diehard - battery of tests, (1997),
<http://stat.fsu.edu/~geo/diehard.html>,
<http://www.helsbreth.org/random/diehard.html>.
10. D. E. Knuth, The art of computer programming, 3-rd Edition, *Addison Wesley*, (1998), pp 27–29, 35, 186–188.
11. A. Folmsbee, AES Java technology comparisons, *Second AES Candidate Conference*, (1999),
<http://csrc.ncsl.nist.gov/encryption/aes/round1/conf2/papers/folmsbee.pdf>.
12. R. L. Rivest, The RC5 encryption algorithm, *Fast Software Encryption: Second International Workshop*, **1008** (1995), pp 86–96,
<http://theory.lcs.mit.edu/~rivest/rc5rev.ps>.
13. R. L. Rivest, M. J. B. Robshaw, R. Sidney, Y. L. Yin, The RC6 block cipher, *The First AES Candidate Conference*, (1998),
<http://theory.lcs.mit.edu/~rivest/rc6.ps>.
14. J. Daemen, V. Rijmen, AES proposal: Rijndael, *The First AES Candidate Conference*, (1998),
<http://www.esat.kuleuven.ac.be/~rijmen/rijndael/rijndaeldocV2.zip>.
15. J. Soto, L. Bassham, Randomness testing of the AES finalist candidates, *The Third AES Candidate Conference*, (2000),
<http://csrc.nist.gov/encryption/aes/round2/conf3/papers/30-jsoto.pdf>.

Speeding up the Arithmetic on Koblitz Curves of Genus Two

Christian Günther¹, Tanja Lange², and Andreas Stein³

¹ Siemens AG, Corporate Technology, Munich, Germany
`christian-c.guenther@mchp.siemens.de`

² Institut für Geometrie, TU Braunschweig, Pockelsstr. 14,
38106 Braunschweig, Germany,
`ta.lange@tu-bs.de`

³ Department of Combinatorics and Optimization,
Centre for Applied Cryptographic Research, University of Waterloo,
Waterloo, ON, Canada N2L 3G1
`astein@cacr.math.uwaterloo.ca`

Abstract. Koblitz, Solinas, and others investigated a family of elliptic curves which admit faster cryptosystem computations. In this paper, we generalize their ideas to hyperelliptic curves of genus 2. We consider the following two hyperelliptic curves $C_\alpha : v^2 + uv = u^5 + \alpha u^2 + 1$ defined over \mathbb{F}_2 with $\alpha = 0, 1$, and show how to speed up the arithmetic in the Jacobian $\mathbb{J}_{C_\alpha}(\mathbb{F}_{2^n})$ by making use of the Frobenius automorphism. With two precomputations, we are able to obtain a speed-up by a factor of 5.5 compared to the generic double-and-add-method in the Jacobian. If we allow 6 precomputations, we are even able to speed up by a factor of 7.

1 Introduction

Public-key cryptosystems based on the discrete logarithm problem on elliptic curves over finite fields have been invented by Neal Koblitz [9] and Victor Miller [17]. Elliptic curve cryptosystems became a popular choice for implementations. The most important operation in an elliptic curve based cryptosystem is the computation of m -folds with a positive integer m . That means computing mP for a point P on an elliptic curve. For example, the complexity of the ElGamal encryption scheme [3] and the Diffie-Hellmann key agreement protocol [2] on an elliptic curve both depend mostly on the complexity of computing m -folds.

The standard method for computing m -folds in a group G is the *double-and-add-method*. If P is an element of G and m a positive integer, doubles and additions are performed with respect to the binary representation of m requiring about $\log_2(m)$ doubles and $\log_2(m)/2$ additions on average. Assuming that doubles and additions have about the same complexity, this method requires $3\log_2(m)/2$ group operations. Allowing precomputations and using memory, various techniques apply to speed up the double-and-add-method (see [8]).

In [11,12,23,15,24], a family of elliptic curves was investigated which allows to speed up the scalar multiplication considerably with the help of the Frobenius

automorphism. They considered the elliptic curves $E : u^2 + uv = v^3 + av^2 + 1$ defined over \mathbb{F}_2 with base field \mathbb{F}_{2^n} , which are called *elliptic Koblitz curves*.

The fastest known attack to the elliptic curve discrete logarithm problem is the parallelized Pollard's rho method [20,22,27]. As noticed in [5,28], the attack time to these curves can be reduced by a factor of $\sqrt{2n}$ which causes one to select slightly larger secure key parameters.

Hyperelliptic curve cryptosystems have been introduced by Neal Koblitz [10] in 1989. Cantor's algorithm [1] provides an effective algorithm for performing the group law in the Jacobian of a hyperelliptic curve.

In this paper, we generalize the ideas for elliptic Koblitz curves to hyperelliptic curves of genus 2. We concentrate on the following two hyperelliptic curves

$$C_\alpha : v^2 + uv = u^5 + \alpha u^2 + 1 \quad (\alpha = 0, 1) ,$$

which are defined over \mathbb{F}_2 and have the base field \mathbb{F}_{2^n} where n is prime. These curves are generalized Koblitz curves of genus 2 and are twists of each other. Furthermore, they are the only non-supersingular curves mentioned in [10] and thus resist the Frey-Rück-attack [4].

We want to point out that the curves C_α have two major advantages. Firstly, the cardinality of the Jacobian of C_α , $\#\mathbb{J}_{C_\alpha}(\mathbb{F}_{2^n})$, can be easily determined for any n , whereas in general computing $\#\mathbb{J}_C(\mathbb{F}_{2^n})$ for a random hyperelliptic curve over \mathbb{F}_{2^n} of genus 2 appears to be difficult. Secondly, we can use the Frobenius automorphism to eliminate many cryptosystem operations. On the cost of two precomputations, we are able to speed up the computation of m -folds by a factor of 5.5. With 6 precomputations, we even obtain a speed-up by a factor of 7. On the other side, a generalization of the methods in [5,28] shows that one can speed up the attack to hyperelliptic cryptosystems by a factor of $\sqrt{2n}$, if the curve has an automorphism of order n (see [7]). Since the curves C_α have at least an automorphism of order n , namely the Frobenius automorphism, the attack to cryptosystems based on the discrete logarithm in $\mathbb{J}_{C_\alpha}(\mathbb{F}_{2^n})$ can be sped up by a factor of $\sqrt{2n}$. As in the case of an elliptic curve, one then has to adjust the size of the key space marginally. Most of the results can be easily generalized to all other genus 2 hyperelliptic curves and, more general, to curves of arbitrary genus [13]. However, for arbitrary curves, one has to be careful with the parameter choice for g and n . Results in [6,7] seem to suggest that hyperelliptic curves of genus $g \geq 4$ are not as secure as elliptic curves.

2 Hyperelliptic Curves

2.1 Basic Definitions

For details on hyperelliptic curves we refer to [10,16,1,26]. Let \mathbb{F}_q be a finite field and $\bar{\mathbb{F}}_q$ its algebraic closure. A non-singular hyperelliptic curve of genus g is defined by the equation

$$C : v^2 + h(u)v = f(u) \quad \text{in } \mathbb{F}_q[u, v] , \quad (2.1)$$

where $h(u), f(u) \in \mathbb{F}_q[u]$, $\deg_u(h) \leq g$, $f(u)$ monic, $\deg_u(f) = 2g + 1$, and if $y^2 + h(x)y = f(x)$ for $(x, y) \in \overline{\mathbb{F}}_q \times \overline{\mathbb{F}}_q$, then $2y + h(x) \neq 0$ or $h'(x)y - f'(x) \neq 0$.

Let \mathbb{F}_{q^n} be a subfield of $\overline{\mathbb{F}}_q$ containing \mathbb{F}_q . The set of \mathbb{F}_{q^n} -points P on C is given by $C(\mathbb{F}_{q^n}) = \{(x, y) \in \mathbb{F}_{q^n}^2 \mid y^2 + h(x)y = f(x)\} \cup \{\infty\}$, where ∞ denotes the point at infinity. For a \mathbb{F}_{q^n} -point $P = (x, y) \in \mathbb{F}_{q^n}^2$, the *opposite* \tilde{P} of P is immediately given by $\tilde{P} = (x, -y - h(x))$. For $P = \infty$ define $\tilde{P} = \infty$.

A *divisor* on C is a finite formal sum $D = \sum_P m_P P$, where m_P are integers that are 0 for almost all P . Then the *degree* of D is defined by $\deg D = \sum_P m_P$. D is said to be *defined over* \mathbb{F}_{q^n} , if¹ $D^\sigma = \sum_P m_P P^\sigma = D$ for any $\sigma \in \text{Aut}(\overline{\mathbb{F}}_q/\mathbb{F}_{q^n})$. The set $\mathbb{D}_C(\mathbb{F}_{q^n})$ of divisors of C defined over \mathbb{F}_{q^n} forms an additive group which contains the finite subgroup $\mathbb{D}_C^0(\mathbb{F}_{q^n})$ of all degree zero divisors of \mathbb{D} defined over \mathbb{F}_{q^n} . The *greatest common divisor* of $D_1 = \sum_{P \in C} m_P P$ and $D_2 = \sum_{P \in C} n_P P$ in $\mathbb{D}_C^0(\mathbb{F}_{q^n})$ is defined by

$$\gcd(D_1, D_2) = \sum_{P \in C} \min(m_P, n_P) P - \left(\sum_{P \in C} \min(m_P, n_P) \right) \infty .$$

Furthermore, the divisor of a polynomial $G(u, v) \in \overline{\mathbb{F}}_q[u, v]$ is defined by $\text{div}(G(u, v)) = \sum_P \text{ord}_P(G) P - \sum_P \text{ord}_P(G) \infty$, where $\text{ord}_P(G)$ is the order of vanishing of $G(u, v)$ at P . Now, the divisor of a rational function $G(u, v)/H(u, v)$ is called a *principal divisor* and is defined by $\text{div}(G(u, v)/H(u, v)) = \text{div}(G(u, v)) - \text{div}(H(u, v))$. We denote by $\mathbb{P}_C(\mathbb{F}_{q^n})$ the group of principal divisors. Since every principal divisor has degree 0, $\mathbb{P}_C(\mathbb{F}_{q^n})$ is a subgroup of $\mathbb{D}_C^0(\mathbb{F}_{q^n})$. Finally, the *Jacobian of C over \mathbb{F}_{q^n}* is given by

$$\mathbb{J}_C(\mathbb{F}_{q^n}) = \mathbb{D}_C^0(\mathbb{F}_{q^n}) / \mathbb{P}_C(\mathbb{F}_{q^n}) .$$

2.2 Reduced Divisors

Let C_f be the set $C - \{\infty\}$ of finite points on C . A degree zero divisor $D = \sum_{P \in C_f} m_P P - \left(\sum_{P \in C_f} m_P \right) \infty$ is called *semi-reduced*, if it satisfies the following conditions for each $P \in C_f$:

- (i) $m_P \geq 0$.
- (ii) If $P \neq \tilde{P}$ and $m_P > 0$, then $m_{\tilde{P}} = 0$.
- (iii) If $P = \tilde{P}$ and $m_P > 0$, then $m_P = 1$.

A semi-reduced divisor is called *reduced*, if in addition

$$\sum_{P \in C_f} m_P \leq g .$$

Reduced divisors have the crucial property (see [19]) that for each degree zero divisor D there exists a unique reduced divisor D_r such that $D - D_r$ is a principal divisor. That means, the set of reduced divisors of C forms a complete system of

¹ P^σ denotes $(\sigma(x), \sigma(y))$, if $P = (x, y) \in \mathbb{F}_{q^n}^2$, and ∞ , if $P = \infty$

representatives for the divisor classes of C . Semi-reduced divisors can be represented uniquely in an advantageous way (see [16,19]): Let $D = \sum_{P \in C_f} m_P P - (\sum_{P \in C_f} m_P) \infty$ be a semi-reduced divisor which is defined over \mathbb{F}_{q^n} . We put $a(u) = \prod_{P \in C_f} (u - x_P)^{m_P} \in \mathbb{F}_{q^n}[u]$, where $P = (x_P, y_P)$. Then there exists a unique polynomial $b(u) \in \mathbb{F}_{q^n}[u]$ such that $D = \gcd(\text{div}(a(u)), (\text{div}(b(u) - v)))$ and

1. $\deg_u b < \deg_u a$,
2. $b(x_P) = y_P$ for each $P \in C_f$ with $m_P \neq 0$,
3. $a(u)$ divides $b(u)^2 + b(u)h(u) - f(u)$.

In this case, we write $D = [a(u), b(u)]$. It follows that every element D of $\mathbb{J}_C(\mathbb{F}_{q^n})$ can be uniquely represented by two polynomials $a(u), b(u) \in \mathbb{F}_{q^n}[u]$, where $a(u)$ is monic, $\deg b(u) < \deg a(u) \leq g$, and $a(u)$ divides $b(u)^2 + b(u)h(u) - f(u)$. We notice that operations in the Jacobian can be performed by using the arithmetic in $\mathbb{F}_{q^n}[u]$. Without explaining the algorithms here, we mention that there exists an effective method to add two elements of the Jacobian which is known as *Cantor's algorithm*. For details we refer to [1,10,16]. The generic operation needs $17g^2 + O(g)$ operations in \mathbb{F}_{q^n} whereas doubling needs $16g^2 + O(g)$ operations in \mathbb{F}_{q^n} (see [25])². So, we can assume that both operations have roughly the same complexity. It is important to note that inversion is basically for free, since the negative of $D = [a(u), b(u)]$ is given by $-D = [a(u), -h(u) - b(u)]$.

2.3 Frobenius Automorphism

The Frobenius automorphism $\phi : \overline{\mathbb{F}}_q \rightarrow \overline{\mathbb{F}}_q, x \mapsto x^q$ extends to an automorphism on the Jacobian of C . Namely, we put $P^\phi = (x^q, y^q)$ for $P = (x, y) \in \overline{\mathbb{F}}_q \times \overline{\mathbb{F}}_q$, and $\infty^\phi = \infty$. For a divisor $D = \sum_{P \in C} m_P P$ of C define D^ϕ to be $\sum_{P \in C} m_P P^\phi$. A very important property of the Frobenius action on semi-reduced divisors is provided by the following

Theorem 2.1. *Let $C : v^2 + h(u)v = f(u)$ be a hyperelliptic curve of genus g , where $h(u), f(u) \in \mathbb{F}_q[u]$. If D is a semi-reduced divisor of C which is defined over \mathbb{F}_{q^n} , then the divisor D^ϕ is semi-reduced and defined over \mathbb{F}_{q^n} . In particular, if $D = [a(u), b(u)]$ with $a(u), b(u) \in \mathbb{F}_{q^n}[u]$, then we have*

$$D^\phi = [a(u)^\phi, b(u)^\phi] .$$

Proof. Let $D = \sum_{P \in C_f} m_P P - (\sum_{P \in C_f} m_P) \infty$ be a semi-reduced divisor which is defined over \mathbb{F}_{q^n} . Since ϕ is an automorphism, each property, that D^ϕ has to satisfy for being semi-reduced, follows from the condition that D is semi-reduced. ϕ commutes with any $\sigma \in \text{Aut}(\overline{\mathbb{F}}_q/\mathbb{F}_{q^n})$. Therefore, we have $(D^\phi)^\sigma = (D^\sigma)^\phi = D^\phi$ for any $\sigma \in \text{Aut}(\overline{\mathbb{F}}_q/\mathbb{F}_{q^n})$. That means D^ϕ is defined over \mathbb{F}_{q^n} . Now, let $a(u) = \prod_{P \in C_f} (u - x_P)^{m_P} \in \mathbb{F}_{q^n}[u]$ and let $b(u)$ be the unique polynomial satisfying a)-c) in Section 2.2 such that

² We remark that there exist even faster methods if the characteristic of \mathbb{F}_{q^n} is 2 and if we use normal basis representation for elements in \mathbb{F}_{q^n}

$$D = \gcd(\operatorname{div}(a(u)), \operatorname{div}(b(u) - v)) = [a(u), b(u)] ,$$

Then

$$D^\phi = \sum_{P \in C_f} m_P P^\phi - \left(\sum_{P \in C_f} m_P \right) \infty = \gcd(\operatorname{div}(\bar{a}(u)), \operatorname{div}(\bar{b}(u) - v)) = [\bar{a}(u), \bar{b}(u)] ,$$

where $\bar{a}(u) = \prod_{P \in C_f} (u - x_P^q)^{m_P} \in \mathbb{F}_{q^n}[u]$, and $\bar{b}(u) \in \mathbb{F}_{q^n}[u]$ is the unique polynomial satisfying a)-c) in Section 2.2 for D^ϕ . Clearly, $a^\phi(u) = \prod_{P \in C_f} (u - x_P^q)^{m_P} = \bar{a}(u)$. It remains to show that $b^\phi(u) = \bar{b}(u)$. Firstly, we have $\deg_u b^\phi < \deg_u a^\phi = \deg_u \bar{a}$, since $\deg_u b < \deg_u a$. Secondly, $b^\phi(x_P^q) = (b(x_P))^q = y_P^q$ for each $P \in C_f$ with $m_P \neq 0$. Thirdly, $a^\phi(u)$ divides $(b(u)^2 + b(u)h(u) - f(u))^\phi = b^\phi(u)^2 + b^\phi(u)h(u) - f(u)$, since $a(u)$ divides $b(u)^2 + b(u)h(u) - f(u)$ and $h(u), f(u) \in \mathbb{F}_q[u]$. Since $\bar{b}(u)$ is unique with these three properties for D^ϕ and $\bar{a}(u) = a^\phi(u)$, we must have $b^\phi(u) = \bar{b}(u)$.

An important consequence of this theorem is that if $D = [a(u), b(u)]$ is a reduced divisor representing an element of $\mathbb{J}_C(\mathbb{F}_{q^n})$, then the action of the Frobenius on D is given by $D^\phi = [a(u)^\phi, b(u)^\phi]$. Notice that this is only true since C is defined over the subfield \mathbb{F}_q of \mathbb{F}_{q^n} . The interpretation is that if $a(u) = \sum_{i=0}^k a_i u^i \in \mathbb{F}_{q^n}[u]$ and $b(u) = \sum_{i=0}^{k'} b_i u^i \in \mathbb{F}_{q^n}[u]$ are the explicit representations of $a(u)$ and $b(u)$, then $a^\phi(u) = \sum_{i=0}^k a_i^q u^i$ and $b^\phi(u) = \sum_{i=0}^{k'} b_i^q u^i$. The practical meaning of this observation is that if we use normal basis representation for elements in \mathbb{F}_{q^n} , then $a^\phi(u)$ and $b^\phi(u)$ can be determined by simply shifting the normal basis representation of each coefficient a_i and b_i in order to compute D^ϕ . The complexity is therefore at most $2g$ cyclic shifts. These shift operations are basically “for free” when compared to the more expensive group operation in the Jacobian.

3 Algorithms for $v^2 + uv = u^5 + \alpha u^2 + 1$

For the remainder of the paper, we consider the curves $C_\alpha : v^2 + uv = u^5 + \alpha u^2 + 1$ with $\alpha = 0, 1$ which are defined over \mathbb{F}_2 . From [10], we know that the characteristic equation of the Frobenius of the curve C_α is given by

$$T^4 + (-1)^\alpha T^3 + (-1)^\alpha 2T + 4 = 0 . \quad (3.2)$$

It follows that

$$-4D = \phi^4(D) + (-1)^\alpha \phi^3(D) + (-1)^\alpha 2\phi(D) \quad (3.3)$$

for any $D \in \mathbb{J}_{C_\alpha}(\overline{\mathbb{F}}_2)$. Here, $\phi(D) := D^\phi$. The equation (3.2) has four solutions

$$\tau_{1/2} = (-1)^{\alpha+1}(\mu_1 \pm i\sqrt{4 - \mu_1})/2 \quad , \quad \tau_{3/4} = (-1)^{\alpha+1}(\mu_2 \pm i\sqrt{4 - \mu_2})/2 \quad ,$$

where $\mu_{1/2} = (1 \pm \sqrt{17})/2$. We put $\tau = \tau_1$ and can regard τ as the element ϕ in the automorphism ring of $\mathbb{J}_{C_1}(\overline{\mathbb{F}}_2)$. As the roots for both curves are equal up to signs, it suffices to consider C_1 . Analogous results hold true for the curve C_0 with some slight modifications. In particular, $\#\mathbb{J}_{C_0}(\mathbb{F}_{2^n})$ differs from $\#\mathbb{J}_{C_1}(\mathbb{F}_{2^n})$ only for odd n .

3.1 Computing τ -Adic Expansions

We are interested in expansions like $11 = -\tau^7 + \tau^4 - 2\tau^2 + 3$, which enable us to compute $11D$ by $11D = -\phi^7(D) + \phi^4(D) - 2\phi^2(D) + 3D$. More generally, for a given integer m , we are interested in its τ -adic expansion $m = \sum_{i=0}^{l-1} c_i \tau^i$ with coefficients $c_i \in R$, where R is a suitable subset of the integers. First, we consider $R = \{0, \pm 1, \pm 2, \pm 3\}$. In Sect. 5, we will vary the set R .

Let $z = a + b\tau + c\tau^2 + d\tau^3$ be any element of $\mathbb{Z}[\tau]$. Since τ is a root of (3.2), z is divisible by τ if and only if $4 \mid a$ as can be shown by direct computation. Therefore, we must have $\tau \mid z - u$ for some $u \in \{0, 1, 2, 3\}$. It follows that

$$z - u = \tau \left(\left(\frac{a-u}{2} + b \right) + c\tau + \left(\frac{a-u}{4} + d \right) \tau^2 - \frac{a-u}{4} \tau^3 \right). \quad (3.4)$$

With $R = \{0, \pm 1, \pm 2, \pm 3\}$ we are able to realize the strategy "at least one of four consecutive coefficients is zero" when determining the c_i 's. The basic algorithm for computing τ -adic expansions of $z = a + b\tau + c\tau^2 + d\tau^3 \in \mathbb{Z}[\tau]$ is to choose an $u \in R$ such that $4 \mid z - u$, to divide $z - u$ by τ and then to repeat these two steps with the new, replaced $z' = ((a-u)/2 + b) + c\tau + ((a-u)/4 + d)\tau^2 - ((a-u)/4)\tau^3$, see (3.4), until the resulting z' will be zero. Then the sequence of those u 's will be the sequence of the coefficients $c_0, \dots, c_{l-1} \in R$ we have searched for. We proceed as follows:

1. If $4 \mid a$, then $\tau \mid z$ and we clearly use $u = 0$.
2. If $4 \nmid a$, then since $R = \{0, \pm 1, \pm 2, \pm 3\}$ we have exactly two choices for u and we can try to make one of the subsequent a 's divisible by 4:
 - (a) If $2 \mid b$, then there is exactly one $u \in R$ such that $4 \mid a - u$ and $4 \mid ((a-u)/2 + b)$, namely

| $b \bmod 4 \backslash a \bmod 8$ | 1 | 2 | 3 | 5 | 6 | 7 |
|----------------------------------|----|----|----|----|----|----|
| 0 | 1 | 2 | 3 | -3 | -2 | -1 |
| 2 | -3 | -2 | -1 | 1 | 2 | 3 |

Using these values for u , the actual u is non zero but the next one will be zero.

- (b) If $2 \nmid b$, then we are only able to make the third successor of the actual a at the latest be divisible by 4 by using.

| $d \bmod 2 \backslash a \bmod 8$ | 1 | 2 | 3 | 5 | 6 | 7 |
|----------------------------------|----|----|----|----|----|----|
| 0 | 1 | 2 | 3 | -3 | -2 | -1 |
| 1 | -3 | -2 | -1 | 1 | 2 | 3 |

This strategy produces expansions $m = \sum_{i=0}^{l-1} c_i \tau^i$ with coefficients c_i in $R = \{0, \pm 1, \pm 2, \pm 3\}$, where $c_i c_{i+1} c_{i+2} c_{i+3} = 0$ ($i \in \{0, \dots, l-4\}$).

For an integer m , the expected length l of such an expansion is $2 \log_2 |m|$. Note that this is about twice as long as the binary expansion $m = \sum b_i 2^i$, where $b_i \in \{0, 1\}$. We will show in the following section how to reduce the length of the τ -adic representation.

3.2 Reducing the Length of the Representation

Since the order of the Frobenius is n , two automorphisms are the same, if $\sum_{i=0}^{l_1-1} c_i \phi^i - \sum_{i=0}^{l_2-1} d_i \phi^i \in (\phi^n - 1)\mathbb{Z}[\phi]$. Thus the corresponding τ -adic expansions are equivalent. Therefore before expanding m , we reduce it modulo $\tau^n - 1$ in $\mathbb{Z}[\tau]$ to obtain a shorter representation $[m] = \sum_{i=0}^{l-1} c_i \phi^i$ of the multiplication-by- m -map. We look for an element $M \in \mathbb{Z}[\tau]$ such that $M \equiv m \pmod{\tau^n - 1}$ and the τ -adic expansion of M is as short as possible, i.e. $|M|$ is as small as possible.

Theorem 3.1. *For any positive integers m and n , there exists an element $M \in \mathbb{Z}[\tau]$ such that*

1. $m \equiv M \pmod{\tau^n - 1}$,
2. $2 \log_2 |M| < n + 5$.

Proof. Let $r = m/(\tau^n - 1) \in \mathbb{Q}(\tau)$. Then there exist r_0, r_1, r_2, r_3 in \mathbb{Q} such that $r = \sum_{i=0}^3 r_i \tau^i$. Let v_i be the nearest integer to r_i for $i = 0, \dots, 3$. We put $v = \sum_{i=0}^3 v_i \tau^i$ and $M = m - v(\tau^n - 1)$. Then $m \equiv M \pmod{\tau^n - 1}$. By using the identity

$$|M|^2 = |m - v(\tau^n - 1)|^2 = \left| \frac{m}{\tau^n - 1} - v \right|^2 |\tau^n - 1|^2$$

we derive that $|M/(\tau^n - 1)|^2 < 14$. Since $|\tau^n - 1|^2 < (2^{n/2} + 1)^2$, we derive that $2 \log_2 |M| < n + 5$.

For any positive integers m and n , we are easily able to determine an element $M \in \mathbb{Z}[\tau]$, satisfying $m \equiv M \pmod{\tau^n - 1}$ and having a τ -adic expansion of length $l \sim n$. We call this representation the *reduced τ -adic expansion* of m . In the automorphism ring of the Jacobian, we obtain for the multiplication-by- m map that $[m] = \sum_{i=0}^{l-1} c_i \phi^i$. The algorithm to compute M from m is along the lines of the proof of Theorem 3.1. We therefore omit it. We remark here that we need to be able to find a representation of $\tau^n - 1$ as $\tau^n - 1 = a + b\tau + c\tau^2 + d\tau^3$ with integers a, b, c, d . The next section will solve this problem. Furthermore, we need to be able to compute multiplicative inverses in $\mathbb{Q}[\tau]$. This can be done by the usual extended gcd for polynomials.

3.3 Representing $\tau^n - 1$ by $a + b\tau + c\tau^2 + d\tau^3$

To compute $a, b, c, d \in \mathbb{Z}$ such that $\tau^n - 1 = a + b\tau + c\tau^2 + d\tau^3$ is no difficult task. Let n be a positive integer. Suppose that $\tau^{n-1} = a_{n-1} + b_{n-1}\tau + c_{n-1}\tau^2 + d_{n-1}\tau^3$ for unique integers $a_{n-1}, b_{n-1}, c_{n-1}, d_{n-1}$, then

$$\begin{aligned} \tau^n &= a_{n-1}\tau + b_{n-1}\tau^2 + c_{n-1}\tau^3 + d_{n-1}\tau^4 \\ &= -4d_{n-1} + (a_{n-1} + 2d_{n-1})\tau + b_{n-1}\tau^2 + (c_{n-1} + d_{n-1})\tau^3, \end{aligned}$$

since $\tau^4 = -4 + 2\tau + \tau^3$, and hence

$$\tau^n - 1 = -(4d_{n-1} + 1) + (a_{n-1} + 2d_{n-1})\tau + b_{n-1}\tau^2 + (c_{n-1} + d_{n-1})\tau^3.$$

Starting with $\tau^0 = 1$, we can compute the integers a, b, c, d iteratively.

3.4 Computing m -Folds Using τ -Adic Expansions

We now present our main algorithm for computing m -folds in the Jacobian of the genus 2 curve $C_1 : v^2 + uv = u^5 + u^2 + 1$ with base field F_{2^n} . Let $D = [(a(u), b(u))] \in \mathbb{J}_{C_1}(\mathbb{F}_{2^n})$, where $\deg_u b < \deg_u a \leq 2$, and $a(u)$ is monic. For instance, if $\deg_u a = 2$, then $a(u) = a_0 + a_1u + u^2$ and $b(u) = b_0 + b_1u$ with coefficients $a_0, a_1, b_0, b_1 \in \mathbb{F}_{2^n}$. If $\deg_u a < 2$, then we even need less coefficients for $a(u)$ and $b(u)$. We assume that the coefficients of $a(u)$ and $b(u)$ are represented with respect to a normal basis of \mathbb{F}_{2^n} over \mathbb{F}_2 .

Algorithm 3.2. (*Computing Scalar Multiples*)

INPUT: $a(u), b(u) \in \mathbb{F}_{2^n}[u]$ such that $D = [a(u), b(u)] \in \mathbb{J}_{C_1}(\mathbb{F}_{2^n})$; $c_0, \dots, c_{l-1} \in \{0, \pm 1, \pm 2, \pm 3\}$ with

$$m \equiv \sum_{i=0}^{l-1} c_i \tau^i \pmod{\tau^n - 1}.$$

OUTPUT: $s(u), t(u) \in \mathbb{F}_{2^n}[u]$ such that $mD = [s(u), t(u)] \in \mathbb{J}_{C_1}(\mathbb{F}_{2^n})$.

1. Precompute $2D$ and $3D$.
2. $H \leftarrow c_{l-1}D = [s(u), t(u)]$;
3. **For** i **from** $l-2$ **downto** 0 **do**:
 - (a) $H \leftarrow H^\phi = [s(u)^\phi, t(u)^\phi]$;
 - (b) **If** $(c_i \neq 0)$ $H \leftarrow H + c_i D$;
4. **Output** $(s(u), t(u))$.

Note that the operation $H = H^\phi$ is nothing else than cyclic shifting of at most 4 coefficients of $s(u)$ and $t(u)$, if $s(u)$ and $t(u)$ are represented with respect to a normal basis. In general, $s(u) = s_0 + s_1u + u^2$, $t(u) = t_0 + t_1u$ with $s_0, s_1, t_0, t_1 \in \mathbb{F}_{2^n}$. Thus, $H^\phi = [s_0^2 + s_1^2u + u^2, t_0^2 + t_1^2u]$, and the computation of H^ϕ needs four cyclic shifts of elements in \mathbb{F}_{2^n} .

3.5 Computing the Cardinality of the Jacobian

The main step in computing the cardinality of $\mathbb{J}_{C_1}(\mathbb{F}_{2^n})$ is to determine the characteristic polynomial of the Frobenius. Since C_1 is defined over \mathbb{F}_2 , this can be accomplished very easily. The Theorem of Weil then gives us immediately that

$$\#\mathbb{J}_{C_1}(\mathbb{F}_{2^n}) = \prod_{i=1}^4 (1 - \tau_i^n) = ((1 + 2^n) - (\tau_1^n + \tau_2^n))((1 + 2^n) - (\tau_3^n + \tau_4^n)).$$

It appears that an explicit formula for the cardinality of Jacobians over \mathbb{F}_{2^n} can only be developed for supersingular curves (see [10]). For our curve, we have to proceed differently. A way of evaluating products of the form $\prod_{i=1}^r (1 - \alpha_i^n)$, where α_i , $i = 1, \dots, r$ are the roots of an arbitrary polynomial, was suggested by Pierce [21] and Lehmer [14]. Our case is very special. We can exploit the

Table 1. Average length and density

| n | average length | average density | n | average length | average density |
|-----|----------------|-----------------|-----|----------------|-----------------|
| 61 | 62.38 | 0.5460 | 97 | 98.34 | 0.5437 |
| 67 | 68.36 | 0.5458 | 101 | 102.36 | 0.5433 |
| 71 | 72.38 | 0.5455 | 103 | 104.31 | 0.5429 |
| 73 | 74.35 | 0.5449 | 107 | 108.33 | 0.5434 |
| 79 | 80.33 | 0.5445 | 109 | 110.34 | 0.5424 |
| 83 | 84.35 | 0.5440 | 113 | 114.35 | 0.5427 |
| 89 | 90.32 | 0.5441 | | | |

additional structure of $P(T)$ to derive recursions of lower order and using integer arithmetic only. If we assume that $\tau_1^n + \tau_2^n = A_n + \mu_1 B_n$, then we get for $n \geq 2$ that $\tau_1^n + \tau_2^n = (4B_{n-1} - 2A_{n-2}) + \mu_1(A_{n-1} + B_{n-1} - 2B_{n-2})$. Equating coefficients yields the following recursions. Put $A_0 = 2$, $A_1 = 0$, $B_0 = 0$, and $B_1 = 1$. For $n \geq 2$ we define $A_n = 4B_{n-1} - 2A_{n-2}$ and $B_n = A_{n-1} + B_{n-1} - 2B_{n-2}$. Then we have

$$\#\mathbb{J}_{C_1}(\mathbb{F}_{2^n}) = (1 + 2^n)^2 - (2A_n + B_n)(1 + 2^n) + (A_n^2 + A_n B_n - 4B_n^2) .$$

A similar approach leads to formulas for $\#\mathbb{J}_{C_0}(\mathbb{F}_{2^n})$.

4 Experimental Results

This section contains three tables. Table 1 describes the length and the density of reduced τ -adic expansions. For each prime $n \in \{61, \dots, 113\}$, we generated 10000 random integers m in the range $0 < m < \#\mathbb{J}_{C_1}(\mathbb{F}_{2^n})$ and computed the reduced τ -adic representation. If d denotes the number of the nonzero coefficients c_i , and l the length of the representation, the quotient d/l is its density.

The value $n + \frac{4}{3}$ seems to be a good approximation for the expected length l of a reduced τ -adic expansion. The asymptotic density (obtained by combinatorial means) is $\frac{489}{910} \sim 0.537$. The experiments provide evidence that the density is approaching the expected bound, so that the number of nonzero coefficients c_i is approximately $\frac{489}{910}(n + \frac{4}{3})$. Therefore, Algorithm 3.2 for computing multiples mD for $D \in \mathbb{J}_{C_1}(\mathbb{F}_{2^n})$ needs about $\frac{5}{9}n$ additions of reduced divisors, while the shift operations are essentially for free. The double-and-add-method for $\mathbb{J}_{C_1}(\mathbb{F}_{2^n})$ needs about $2n$ doubles and n additions of reduced divisors, so that the τ -adic method leads to a speed-up by a factor of

$$\frac{3n}{\frac{489n}{910}} \sim 5.5 .$$

In Table 2 and 3, respectively, we list examples for factorizations of $\#J_{C_1}(\mathbb{F}_{2^n})$ and $\#J_{C_0}(\mathbb{F}_{2^n})$. We only considered the cases where n takes on prime values in the range 61 to 113 and where the cardinalities of Jacobians contain a large prime factor.

Table 2. Computing the cardinality of the Jacobian $\mathbb{J}_{C_1}(\mathbb{F}_{2^n})$

| n | $\#\mathbb{J}_{C_1}(\mathbb{F}_{2^n})$ |
|-----|---|
| 61 | 5316911976894487061973100640561324954 = $2 \cdot 2658455988447243530986550320280662477$ |
| 67 | 21778071481105140023832236795388122729642 = $2 \cdot 3217 \cdot 3384841697405212935006564624710619013$ |
| 97 | 25108406941546737996390354885625124943376439570684227477754 = $2 \cdot 389 \cdot 1747 \cdot 18473392463868826910318794676754071940716909907019619$ |
| 103 | 102844034832575383397207943835010553634640254575820398436691978 = $2 \cdot 47381 \cdot 1085287719049570327739050925845914539948927360923370110769$ |
| 109 | 421249166674228800251100330124945140261321879842750041189776992282 = $2 \cdot 2617 \cdot 620764811 \cdot 129651709107106280529021406475320711149271787278988543$ |
| 113 | 10783978668602557431646595347682461521285605430038087099528386736762 = $2 \cdot 53919893334301278715823297673841230760642802715019043549764193368381$ |

Table 3. Computing the cardinality of the Jacobian $\mathbb{J}_{C_0}(\mathbb{F}_{2^n})$

| n | $\#\mathbb{J}_{C_0}(\mathbb{F}_{2^n})$ |
|-----|---|
| 67 | 21778071484774983299499715182968742769496 = $2^3 \cdot 2722258935596872912437464397871092846187$ |
| 89 | 383123885216451157219690382614340814499889612946264008 = $2^3 \cdot 179 \cdot 1069 \cdot 83091469 \cdot 301204924452353711515420284982459139979$ |

5 Improvements

Following the idea of Koblitz [12], we modified our set of possible coefficients and used the set

$$R' = \{0, \pm 1, \pm 2, \pm(1 + \tau), \pm(1 - \tau), \pm(1 - 2\tau), \pm 2 + \tau\}$$

as the domain of coefficients. Accepting the cost of 6 precomputations and storing these elements (instead of only 2 for set R), this choice enables us to realize a sparse τ -adic expansion in the sense that no two consecutive coefficients are nonzero (cf. [24]). Using u as in the following table we force $a + b\tau + c\tau^2 + d\tau^3 - u$ to be divisible by τ^2 , i.e. the next coefficient will be zero. If $4|a$ then $u = 0$, else take

| $b \bmod 4 \backslash a \bmod 8$ | 1 | 2 | 3 | 5 | 6 | 7 |
|----------------------------------|-------------|-------------|----------------|-------------|-------------|----------------|
| 0 | 1 | 2 | $-(1 - 2\tau)$ | $1 - 2\tau$ | -2 | -1 |
| 1 | $1 + \tau$ | $2 + \tau$ | $-(1 + \tau)$ | $1 - \tau$ | $-2 + \tau$ | $-(1 - \tau)$ |
| 2 | $1 - 2\tau$ | -2 | -1 | 1 | 2 | $-(1 - 2\tau)$ |
| 3 | $1 - \tau$ | $-2 + \tau$ | $-(1 - \tau)$ | $1 + \tau$ | $2 + \tau$ | $-(1 + \tau)$ |

By using this modified version of the τ -adic expansion, the average length of the reduced τ -adic representations was $< n + 2$ for an extension of degree n . The expected density for this set of coefficients is $\frac{3}{7} \sim 0.42857$. In Table 4, we present our experimental results. The generation of the integers m was identical to the one in Table 1. The difference lies in the choice of the set R' which yields new τ -adic expansions.

Table 4. Average length and density

| n | average length | average density | n | average length | average density |
|-----|----------------|-----------------|-----|----------------|-----------------|
| 61 | 63.02 | 0.4284 | 97 | 99.67 | 0.4177 |
| 67 | 69.00 | 0.4275 | 101 | 102.95 | 0.4287 |
| 71 | 72.98 | 0.4288 | 103 | 104.93 | 0.4289 |
| 73 | 32.15 | 0.4287 | 107 | 109.05 | 0.4288 |
| 79 | 81.01 | 0.4287 | 109 | 111.01 | 0.4287 |
| 83 | 84.99 | 0.4286 | 113 | 114.96 | 0.4285 |
| 89 | 91.00 | 0.4288 | | | |

Therefore with this set R' we obtain a speed-up by a factor of

$$\frac{3n}{\frac{3n}{7}} = 7$$

with respect to the binary expansion on the cost of more storing and precomputations.

References

1. Cantor, D. G.: Computing in the Jacobian of a Hyperelliptic Curve. *Math. Comp.* **48** (1987) 95–101
2. Diffie, W., Hellman, M. E.: New Directions in Cryptography. *IEEE Trans. Inform. Theory* **22** (1976) 644–654
3. ElGamal, T.: A Public-Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms. *IEEE Trans. Inform. Theory* **31** (1985) 469–472
4. Frey, G., Rück, H.-G.: A Remark Concerning m -Divisibility and the Discrete Logarithm Problem in the Divisor Class Group of Curves. *Math. Comp.* **62** (1994) 865–874
5. Gallant, R., Lambert, R., Vanstone, S.: Improving the Parallelized Pollard Lambda Search on Binary Anomalous Curves. To Appear in *Math. Comp.* <http://www.certicom.com/chal/download/paper.ps>
6. Gaudry, P., Hess, F., Smart, N.: Constructive and Destructive Facets of Weil Descent on Elliptic Curves, preprint, 1999.
7. Gaudry, P., Morain, F., Duursma, I.: Speeding Up the Discrete Log Computation on Curves with Automorphisms In: *Proc. of The Mathematics of Public Key Cryptography*. Fields-Institute Toronto (1999)
8. Gordon, D.: A Survey of Fast Exponentiation Methods. *J. Algorithms* **27** (1998) 129–146
9. Koblitz, N.: Elliptic Curve Cryptosystems. *Math. Comp.* **48** (1987) 203–209
10. Koblitz, N.: Hyperelliptic Cryptosystems. *J. Cryptology* **1** (1989) 139–150
11. Koblitz, N.: CM Curves with Good Cryptographic Properties. In: *Advances in Cryptology – Crypto '91*. LNCS, Vol. 576. Springer-Verlag, Berlin Heidelberg New York (1992) 279–287
12. Koblitz, N.: An Elliptic Curve Implementation of the Finite Field Digital Signature Algorithm. In: *Advances in Cryptology – Crypto '98*. Lecture Notes in Computer Science, Vol. 1462. Springer-Verlag, Berlin Heidelberg New York (1998) 327–337

13. Lange, T.: Efficient Arithmetic on Hyperelliptic Koblitz Curves. preprint, 2000.
14. Lehmer, D.H.: Factorization of Certain Cyclotomic Functions. *Ann. Math.* **34**(1933) 461-479
15. Meier, W., Staffelbach, O.: Efficient Multiplication on Certain Nonsupersingular Elliptic Curves. In: *Advances in Cryptology – Crypto '92*. LNCS, Vol. 740. Springer-Verlag, Berlin Heidelberg New York (1993) 333-344
16. Menezes, A., Wu, Y., Zuccherato, R.: An Elementary Introduction to Hyperelliptic Curves. In: Koblitz, N.: *Algebraic Aspects of Cryptography*. Springer-Verlag, Berlin Heidelberg New York (1998)
17. Miller, V.: Use of Elliptic Curves in Cryptography. In: *Advances in Cryptology – Crypto '85*. LNCS, Vol. 218. Springer-Verlag, Berlin Heidelberg New York (1986) 417-426
18. Müller, V., Stein, A., Thiel, C.: Computing Discrete Logarithms in Real Quadratic Congruence Function Fields of Large Genus. *Math. Comp.* **68** (1999) 807-822
19. Mumford, D.: *Tata Lectures on Theta I, II*. Birkhäuser-Verlag, Boston (1983/84)
20. van Oorschot, P., Wiener, M. J.: Parallel Collision Search with Cryptanalytic Applications. *J. Cryptology* **12** (1999) 1-28
21. Pierce, T.A.: The Numerical Factors of the Arithmetic Forms $\prod_{i=1}^n (1 \pm \alpha_i^m)$. *Ann. Math.* **18**(1916), 53-64.
22. Pollard, J. M.: Kangaroos, Monopoly and Discrete Logarithms. To appear in *J. Cryptology*.
23. Solinas, J.: An Improved Algorithm for Arithmetic on a Family of Elliptic Curves. In: *Advances in Cryptology – Crypto '97*. LNCS, Vol. 1294. Springer-Verlag, Berlin Heidelberg New York (1997) 357-371
24. Solinas, J.: Efficient Arithmetic on Koblitz Curves. Techn. Report CORR 99-09, University of Waterloo (1999), 61 pages. <http://www.cacr.math.uwaterloo.ca>
25. Stein, A.: Sharp Upper Bounds for Arithmetics in Hyperelliptic Function Fields. Techn. Report CORR 99-23, University of Waterloo (1999), 68 pages. Available at <http://www.cacr.math.uwaterloo.ca>
26. Stichtenoth, H.: *Algebraic Function Fields and Codes*. Springer-Verlag, Berlin Heidelberg New York (1993)
27. Teske, E.: Speeding up Pollard's rho method for computing discrete logarithms. In: *Algorithmic Number Theory Seminar ANTS-III*. LNCS, Vol. 1423. Springer-Verlag, Berlin Heidelberg New York (1998) 541-554
28. Wiener, M., Zuccherato, R.: Faster Attacks on Elliptic Curve Cryptosystems. In: *Proceedings of SAC, Workshop on Selected Areas in Cryptography*. LNCS, Springer-Verlag, Berlin Heidelberg New York (1998).

On Complexity of Polynomial Basis Squaring in \mathbb{F}_{2^m}

Huapeng Wu

The Centre for Applied Cryptographic Research,
Department of Combinatorics and Optimization, University of Waterloo, Waterloo,
Canada

`h3wu@cacr.math.uwaterloo.ca`

Abstract. In this paper, the complexity of a squaring operation using polynomial basis (PB) in a class of finite fields \mathbb{F}_{2^m} is evaluated. The main results are as follows:

1. When the field is generated with an irreducible trinomial $f(x) = x^m + x^k + 1$, $1 \leq k \leq \frac{m}{2}$, where both m and k are odd, a PB squaring operation requires $\frac{m-1}{2}$ bit operations.
2. When the field is generated with an irreducible trinomial $f(x) = x^m + x^k + 1$, $1 \leq k \leq \frac{m}{2}$, where $m+k$ is odd and $k \neq \frac{m}{2}$, a PB squaring operation requires $\frac{m+k-1}{2}$ bit operations.
3. When the field is generated with an irreducible trinomial $f(x) = x^m + x^{\frac{m}{2}} + 1$, a PB squaring operation requires $\frac{m+2}{4}$ bit operations.

1 Introduction

Finite field arithmetic has recently been paid much attention mainly because its use in elliptic curve cryptography. In implementing an elliptic curve cryptosystem, a normal basis is usually utilized, because squaring operation in normal basis is only a cyclic shift of the element's coefficients. A multiplication operation can also be performed efficiently with an optimal normal basis (ONB) [5]. It has been shown that a bit-parallel multiplication in \mathbb{F}_{2^m} can be done in about $2m^2$ ground field operations if a type-I ONB is chosen [2]. However, type-I ONB exists only in a small class of fields \mathbb{F}_{2^m} where m is an even number. Moreover, it is more likely to have a comparatively efficient discrete elliptic curve logarithm when m is composite [4]. On the other hand, it has been shown that a bit-parallel multiplier using trinomial-based polynomial basis (TPB) has about the same complexity as that using a type-I ONB [3], while irreducible trinomial over \mathbb{F}_{2^m} exists much more prevalingly than type-I ONB. A squaring operation in TPB, however, is not free.

In this short article, we derive the complexity of a bit-parallel squaring operation using a TPB in \mathbb{F}_{2^m} . It is shown to be of order $O(m)$ ground field operations (comparing to $2m^2$ ground field operation needed for a bit-parallel multiplication operation). If we try to solve an inverse in \mathbb{F}_{2^m} using the method from Fermat theorem, then the complexity of $m-1$ bit-parallel squaring operations

required is not greater than that of half bit-parallel multiplication operation. The time propagation of the hardware architecture of a bit-parallel squarer is also addressed.

The main results include: When the field is generated with an irreducible trinomial $f(x) = x^m + x^k + 1$, $1 \leq k \leq \frac{m}{2}$, then a PB squaring operation requires at most

1. $\frac{m-1}{2}$ bit addition, if both m and k are odd;
2. $\frac{m+k-1}{2}$ bit operations, if $m+k$ is odd and $k \neq \frac{m}{2}$.
3. $\frac{k+1}{2}$ bit operations, if $k = \frac{m}{2}$.

The organization of this paper is as follows: An brief introduction to PB squaring operation is given in Section 2. In Section 3, we present new complexity upper bound for PB squaring operation in a class of finite fields. Hardware bit-parallel implementation is addressed in Section 4. Finally, a few concluding remarks are given in Section 5.

2 Polynomial Basis Squaring Operation

Let $f(x)$ be the irreducible polynomial over \mathbb{F}_2 generating the field \mathbb{F}_{2^m} . Let

$A(x) = \sum_{i=0}^{m-1} a_i x^i$ be the polynomial representation of an arbitrary element of \mathbb{F}_{2^m} . The squaring operation of $A(x)$ is

$$\begin{aligned} C(x) &\triangleq \sum_{i=0}^{m-1} c_i x^i = A^2(x) \bmod f(x) \\ &= a_0 + a_1 x^2 + a_2 x^4 + \dots + a_{m-1} x^{2m-2} \bmod f(x). \end{aligned}$$

It can be seen that squaring in \mathbb{F}_{2^m} is actually a case of polynomial modular reduction. Then the following corollary is obvious from the results on complexity of polynomial modular reduction [6].

Corollary 1. Let the field \mathbb{F}_{2^m} be generated with the irreducible r -term polynomial $f(x)$ of degree m . Then squaring a field element in parallel can be performed with at most $(r-1)(m-1)$ addition operations in \mathbb{F}_2 .

When $f(x)$ is chosen as an irreducible trinomial, however, the complexity can be further reduced.

3 Complexity Upper Bound for PB Squaring

In this section, we assume that the field is generated with an irreducible trinomial $f(x) = x^m + x^k + 1$, $1 \leq k \leq \frac{m}{2}$. Based on the the parity of m and k , the derivation is divided into the following three cases:

1. Both m and $1 \leq k < \frac{m}{2}$ are odd;
2. m is odd and $1 < k < \frac{m}{2}$ is even;
3. m is even and $1 \leq k \leq \frac{m}{2}$ is odd.

3.1 Both m and $1 \leq k < \frac{m}{2}$ Are Odd

Let

$$A^2(x) = \sum_{i=0}^{m-1} a_i x^{2i} = \sum_{i=0}^{2m-2} a'_i x^i,$$

where $a'_i \triangleq a_{\frac{i}{2}}$ if i even, and 0 if i odd. Define

$$\sum_{i=0}^{m+2l+1} a'_i x^i \bmod f(x) \triangleq \sum_{i=0}^{m-1} t_i^{(l)} x^i,$$

for $l = -1, 0, 1, \dots, \frac{m-1}{2} - 1$. Then we have

$$\sum_{i=0}^{m-1} t_i^{(l)} x^i = \sum_{i=0}^{m-1} t_i^{(l-1)} x^i + a'_{m+2l+1} x^{m+2l+1} \bmod f(x). \quad (1)$$

The coefficient $t_i^{(l)}$'s have their initial values $t_i^{(-1)} = a'_i$, and we try to solve the final values $t_i^{(\frac{m-1}{2}-1)} = c_i, i = 0, 1, \dots, m-1$. Note that $t_i^{(-1)} = 0$ if i is an odd number.

When $l = 0$,

$$\begin{aligned} \sum_{i=0}^{m-1} t_i^{(0)} x^i &= \sum_{i=0}^{m-1} a'_i x^i + a'_{m+1} x^{m+1} \bmod f(x) \\ &= \sum_{i=0}^{m-1} a'_i x^i + a'_{m+1} (x + x^{k+1}) \bmod f(x). \end{aligned}$$

Then we have

$$t_i^{(0)} = \begin{cases} a'_i + a'_{m+1}, & i = k+1; \\ a'_i, & i \text{ even, and } i \neq k+1; \\ a'_{m+1}, & i = 1; \\ 0, & \text{otherwise.} \end{cases}$$

Clearly, one bit addition is needed to compute $t_i^{(0)}$ from $t_i^{(-1)}, i = 0, 1, \dots, m-1$.

In the following we will repeatedly use (1) for $l = 1, 2, \dots, \frac{m}{2} - 1$. It will be seen that there are a few newly generated terms at each step. For example, when $l = 0$ we have two newly generated terms $a'_m x$ and $a'_m x^{k+1}$. Note that $k+1$ is an even number and one bit operation is needed to take care of this even power term. In fact, one bit addition is *always* required if an *even* power term is generated, while one bit operation is *probably* needed if an *odd* power term is generated. This is because for some l , $t_i^{(l-1)}$ could be zero for some odd i .

For $l > 0$ and $l \leq \frac{m-k}{2} - 1$ (in order to keep $k+2l+1 < m$), we have

$$\begin{aligned}
\sum_{i=0}^{m-1} t_i^{(l)} x^i &= \sum_{i=0}^{m-1} t_i^{(l-1)} x^i + a'_{m+2l+1} x^{m+2l+1} \bmod f(x) \\
&= \sum_{i=0}^{m-1} t_i^{(l-1)} x^i + a'_{m+2l+1} x^{2l+1} (1 + x^k) \bmod f(x) \\
&= \sum_{i=0}^{m-1} t_i^{(l-1)} x^i + a'_{m+2l+1} x^{2l+1} + a'_{m+2l+1} x^{k+2l+1} \bmod f(x).
\end{aligned}$$

Obviously in this step (from $l-1$ to l), one odd power term (x^{2l+1}) and one even power term (x^{k+2l+1}) are generated at the right side of the above equation. When l runs through from 0 to $\frac{m-k}{2} - 1$, the value of $2l+1$ runs through the odd numbers from 1 to $m-k-1$, and the value of $k+2l+1$ runs through the even numbers from $k+1$ to $m-1$.

Therefore, when $0 \leq l \leq \frac{m-k}{2} - 1$, we have

$$\begin{aligned}
t_i^{(l)} &= \begin{cases} t_i^{(l-1)} + a'_{m+2l+1}, & i = 2l + k + 1; \\ a'_{m+2l+1}, & i = 2l + 1; \\ t_i^{(l-1)}, & i \text{ even and } i \neq 2l + k + 1; \text{ or } i = 1, 3, \dots, 2l - 1; \\ 0, & \text{otherwise.} \end{cases} \\
&= \begin{cases} t_i^{(l-1)} + a'_{m-k+i}, & i = 2l + k + 1; \\ a'_{m+i}, & i = 2l + 1; \\ t_i^{(l-1)}, & i \text{ even and } i \neq 2l + k + 1; \text{ or } i = 1, 3, \dots, 2l - 1; \\ 0, & \text{otherwise.} \end{cases} \\
&= \begin{cases} a'_i + a'_{m-k+i} & i = k + 1, k + 3, \dots, k + 2l + 1; \\ a'_{m+i} & i = 1, 3, \dots, 2l + 1; \\ a'_i & i \text{ even and } i \neq k + 1, k + 3, \dots, k + 2l + 1; \\ 0 & \text{Otherwise.} \end{cases}
\end{aligned}$$

Thus for $l = \frac{m-k}{2} - 1$, we can solve $t_i^{(l)}$ as follows

$$t_i^{(\frac{m-k}{2}-1)} = \begin{cases} a'_i + a'_{m-k+i} & i = k + 1, k + 3, \dots, m - 1; \\ a'_{m+i} & i = 1, 3, \dots, m - k - 1; \\ a'_i & i = 0, 2, \dots, k - 1; \\ 0 & i = m - k + 1, m - k + 3, \dots, m - 2. \end{cases}$$

In the following, we consider two cases:

1. If $k = 1$.

When $k = 1$, we have $\frac{m-k}{2} - 1 = \frac{m-1}{2} - 1$. Therefore,

$$c_i = t_i^{(\frac{m-1}{2}-1)} = \begin{cases} a'_i + a'_{m-1+i} & i = 2, 4, \dots, m - 1; \\ a'_{m+i} & i = 1, 3, \dots, m - 2; \\ a'_i & i = 0. \end{cases} \quad (2)$$

It can be seen from the (2) that $\frac{m-1}{2}$ bit additions are required for obtaining c_i , $i = 0, 1, \dots, m - 1$.

2. If $1 < k < \frac{m}{2}$.

When $\frac{m-k}{2} \leq l \leq \frac{m-1}{2} - 1$, we have

$$\begin{aligned}
\sum_{i=0}^{m-1} t_i^{(l)} x^i &= \sum_{i=0}^{m-1} t_i^{(l-1)} x^i + a'_{m+2l+1} x^{m+2l+1} \bmod f(x) \\
&= \sum_{i=0}^{m-1} t_i^{(l-1)} x^i + a'_{m+2l+1} x^{2l+1} [1 + x^k] \bmod f(x) \\
&= \sum_{i=0}^{m-1} t_i^{(l-1)} x^i + a'_{m+2l+1} x^{2l+1} + a'_{m+2l+1} x^{2l+k+1} \bmod f(x) \\
&= \sum_{i=0}^{m-1} t_i^{(l-1)} x^i + a'_{m+2l+1} x^{2l+1} \\
&\quad + a'_{m+2l+1} x^{2l+k+1-m} [1 + x^k] \bmod f(x) \\
&= \sum_{i=0}^{m-1} t_i^{(l-1)} x^i + a'_{m+2l+1} x^{2l+1} + a'_{m+2l+1} x^{2l+k+1-m} \\
&\quad + a'_{m+2l+1} x^{2l+2k+1-m} \bmod f(x)
\end{aligned}$$

Since $k \leq \frac{m}{2}$ and $l \leq \frac{m-1}{2} - 1$, we have $2l + 2k + 1 - m \leq m - 2$. It can be seen that there are two newly generated odd power terms (x^{2l+1} and $x^{2l+k+1-m}$) and one even power term ($x^{2l+2k+1-m}$) in this step. When l runs through from $\frac{m-k}{2}$ to $\frac{m-1}{2} - 1$, the value of $2l + 1$ runs through the odd numbers from $m - k + 1$ to $m - 2$, the value of $2l + k + 1 - m$ runs through the odd numbers from 1 to $k - 2$, and the value of $2l + 2k + 1 - m$ runs through the even numbers from $k + 1$ to $2k - 2$.

Therefore, $t_i^{(l)}$ can be given as follows

$$\begin{aligned}
t_i^{(l)} &= \begin{cases} a'_{m+2l+1}, & i = 2l + 1; \\ t_i^{(l-1)} + a'_{m+2l+1}, & i = 2l + 2k + 1 - m, 2l + k + 1 - m; \\ t_i^{(l-1)}, & i \text{ even and } i \neq k + 1, k + 3, \dots, 2l + 2k - 1 - m; \\ & \text{or } i \text{ odd and } i = 1, 3, \dots, 2l + k - 1 - m, \\ & 2l + k + 3 - m, 2l + k + 5 - m, \dots, 2l - 1; \\ 0, & \text{otherwise.} \end{cases} \\
&= \begin{cases} a'_{m+i}, & i = 2l + 1; \\ t_i^{(l-1)} + a'_{2m-k+i}, & i = 2l + k + 1 - m; \\ t_i^{(l-1)} + a'_{2m-2k+i}, & i = 2l + 2k + 1 - m; \\ t_i^{(l-1)}, & i \text{ even and } i \neq k + 1, k + 3, \dots, 2l + 2k - 1 - m; \\ & \text{or } i \text{ odd and } i = 1, 3, \dots, 2l + k - 1 - m, \\ & 2l + k + 3 - m, 2l + k + 5 - m, \dots, 2l - 1; \\ 0, & \text{otherwise.} \end{cases}
\end{aligned}$$

$$= \begin{cases} a'_{m+i} & i = 2l + k + 3 - m, 2l + k + 5 \\ & -m, \dots, 2l + 1; \\ a'_{m+i} + a'_{2m-k+i} & i = 1, 3, \dots, 2l + k + 1 - m; \\ a'_i + a'_{m-k+i} + a'_{2m-2k+i} & i = k + 1, k + 3, \dots, 2l + 2k + 1 - m; \\ a'_i + a'_{m-k+i} & i = 2l + 2k + 3 - m, 2l + 2k + 5 \\ & -m, \dots, m - 1; \\ a'_i & i = 0, 2, \dots, k - 1; \\ 0 & i = 2l + 3, 2l + 5, \dots, m - 2. \end{cases}$$

When $l = \frac{m-1}{2} - 1$, it follows from the above equations

$$c_i = t_i^{(\frac{m-3}{2})} = \begin{cases} a'_{m+i} & i = k, k + 2, \dots, m - 2; \\ a'_{m+i} + a'_{2m-k+i} & i = 1, 3, \dots, k - 2; \\ a'_i + a'_{m-k+i} + a'_{2m-2k+i} & i = k + 1, k + 3, \dots, 2k - 2; \\ a'_i + a'_{m-k+i} & i = 2k, 2k + 2, \dots, m - 1; \\ a'_i & i = 0, 2, \dots, k - 1. \end{cases}$$

Rewrite the above equation as the following

$$c_i = a'_i \quad i = 0, 2, \dots, k - 1; \quad (3a)$$

$$c_i = (a'_{m+i} + a'_{2m-k+i}) \quad i = 1, 3, \dots, k - 2; \quad (3b)$$

$$c_{k+i} = a'_{m+k+i} \quad i = 0, 2, \dots, m - k - 2; \quad (3c)$$

$$c_{k+i} = a'_{k+i} + (a'_{m+i} + a'_{2m-k+i}) \quad i = 1, 3, \dots, k - 2; \quad (3d)$$

$$c_{2k+i} = a'_{2k+i} + a'_{m+k+i} \quad i = 0, 2, \dots, m - 2k - 1; \quad (3e)$$

Then it can be seen from (3b) and (3d) that some partial sums can be reused (indicated with the bracket). This will save $\frac{k-1}{2}$ bit operations. The total number of bit operations required for the squaring operation can be counted from (3a-3e) and it is $\frac{m-1}{2}$.

3.2 m Is Odd and $1 < k < \frac{m}{2}$ Is Even

The definitions of a'_i and $t_i^{(l)}$ are the same as these in the last subsection. We rewrite the equation (1) here for convenience.

$$\sum_{i=0}^{m-1} t_i^{(l)} x^i = \sum_{i=0}^{m-1} t_i^{(l-1)} x^i + a'_{m+2l+1} x^{m+2l+1} \bmod f(x).$$

The terms $t_i^{(l)}$'s have their initial values $t_i^{(-1)} = a'_i$, and we try to solve the final values $t_i^{(\frac{m-1}{2}-1)} = c_i, i = 0, 1, \dots, m - 1$.

When $l = 0$,

$$\sum_{i=0}^{m-1} t_i^{(0)} x^i = \sum_{i=0}^{m-1} a'_i x^i + a'_{m+1} x^{m+1} \bmod f(x)$$

$$= \sum_{i=0}^{m-1} a'_i x^i + a'_{m+1} (x + x^{k+1}) \bmod f(x).$$

It follows

$$t_i^{(0)} = \begin{cases} a'_{m+1}, & i = 1, k+1; \\ a'_i, & i \text{ even}; \\ 0, & i \text{ odd and } i \neq 1, k+1; \end{cases}$$

Since both the newly generated terms are odd power ones, no bit addition is needed to obtain $t_i^{(0)}$ from $t_i^{(-1)}$, $i = 0, 1, \dots, m-1$.

For $l \geq 0$, we have

$$\begin{aligned} \sum_{i=0}^{m-1} t_i^{(l)} x^i &= \sum_{i=0}^{m-1} t_i^{(l-1)} x^i + a'_{m+2l+1} x^{m+2l+1} \\ &= \sum_{i=0}^{m-1} t_i^{(l-1)} x^i + a'_{m+2l+1} x^{2l+1} (1 + x^k) \\ &= \sum_{i=0}^{m-1} t_i^{(l-1)} x^i + a'_{m+2l+1} x^{2l+1} + a'_{m+2l+1} x^{k+2l+1}. \end{aligned}$$

It can be seen that two odd power terms are generated at the right side of the above equation.

When l runs through from 0 to $\frac{k}{2} - 1$, the value of $2l+1$ runs through the odd numbers from 1 to $k-1$, and the value of $k+2l+1$ runs through the odd numbers from $k+1$ to $2k-1$. Note that $2k-1 < m-1$.

Then we have

$$\begin{aligned} t_i^{(l)} &= \begin{cases} a'_{m+2l+1}, & i = 2l+1, k+2l+1; \\ t_i^{(l-1)}, & i \text{ even, or } i \text{ odd and} \\ & i \neq 1, 3, \dots, 2l+1, k+1, k+3, \dots, k+2l-1; \\ 0, & \text{otherwise.} \end{cases} \\ &= \begin{cases} a'_{m+i}, & i = 2l+1; \\ a'_{m-k+i}, & i = k+2l+1; \\ t_i^{(l-1)}, & i \text{ even, or } i \text{ odd and} \\ & i \neq 1, 3, \dots, 2l+1, k+1, k+3, \dots, k+2l-1; \\ 0, & \text{otherwise.} \end{cases} \\ &= \begin{cases} a'_{m+i}, & i = 1, 3, \dots, 2l+1; \\ a'_{m-k+i}, & i = k+1, k+3, \dots, k+2l+1; \\ a'_i, & i = 0, 2, \dots, m-1; \\ 0, & \text{otherwise.} \end{cases} \quad (4) \end{aligned}$$

When $l = \frac{k}{2} - 1$, from the equation (4) we have

$$t_i^{(\frac{k}{2}-1)} = \begin{cases} a'_{m+i}, & i = 1, 3, \dots, k-1; \\ a'_{m-k+i}, & i = k+1, k+3, \dots, 2k-1; \\ a'_i, & i = 0, 2, \dots, m-1; \\ 0, & \text{otherwise.} \end{cases} \quad (5)$$

In the following we consider two cases:

1. If $2k < m-1$.

In this case we have $k \leq m-k-3$. When l runs through from $\frac{k}{2}$ to $\frac{m-k-3}{2}$ (in order to satisfy $k+2l+1 < m-1$), the value of $2l+1$ runs through the odd numbers from $k+1$ to $m-k-2$, and the value of $k+2l+1$ runs through the odd numbers from $2k+1$ to $m-2$.

From (4) and since $2l+1 \geq k+1$, we have

$$\begin{aligned} t_i^{(l)} &= \begin{cases} t_i^{(l-1)} + a'_{m+2l+1}, & i = 2l+1; \\ a'_{m+2l+1}, & i = k+2l+1; \\ t_i^{(l-1)}, & i \text{ even, or } i \text{ odd and} \\ & i = 1, 3, \dots, 2l-1, 2l+3, \dots, k+2l-1; \\ 0, & \text{otherwise.} \end{cases} \\ &= \begin{cases} t_i^{(l-1)} + a'_{m+i}, & i = 2l+1; \\ a'_{m-k+i}, & i = k+2l+1; \\ t_i^{(l-1)}, & i \text{ even, or } i \text{ odd and} \\ & i = 1, 3, \dots, 2l-1, 2l+3, \dots, k+2l-1; \\ 0, & \text{otherwise.} \end{cases} \\ &= \begin{cases} a'_{m+i}, & i = 1, 3, \dots, k-1; \\ a'_{m-k+i}, & i = 2l+3, 2l+5, \dots, k+2l+1; \\ a'_{m+i} + a'_{m-k+i}, & i = k+1, k+3, \dots, 2l+1; \\ a'_i, & i = 0, 2, \dots, m-1; \\ 0, & \text{otherwise.} \end{cases} \end{aligned}$$

When $l = \frac{m-k-3}{2}$, it follows

$$t_i^{(\frac{m-k-3}{2})} = \begin{cases} a'_{m+i}, & i = 1, 3, \dots, k-1; \\ a'_{m+i} + a'_{m-k+i}, & i = k+1, k+3, \dots, m-k-2; \\ a'_{m-k+i}, & i = m-k, m-k+2, \dots, m-2; \\ a'_i, & i = 0, 2, \dots, m-1; \\ 0, & \text{otherwise.} \end{cases}$$

When $\frac{m-k-1}{2} \leq l \leq \frac{m-1}{2} - 1$, we have

$$\begin{aligned} \sum_{i=0}^{m-1} t_i^{(l)} x^i &= \sum_{i=0}^{m-1} t_i^{(l-1)} x^i + a_{m+2l+1} x^{m+2l+1} \\ &= \sum_{i=0}^{m-1} t_i^{(l-1)} x^i + a'_{m+2l+1} x^{2l+1} + a'_{m+2l+1} x^{2l+k+1} \end{aligned}$$

$$\begin{aligned}
&= \sum_{i=0}^{m-1} t_i^{(l-1)} x^i a'_{m+2l+1} x^{2l+1} + a'_{m+2l+1} x^{2l+k+1-m} \\
&\quad + a'_{m+2l+1} x^{2l+2k+1-m}
\end{aligned}$$

When l runs through from $\frac{m-k-1}{2}$ to $\frac{m-1}{2} - 1$, the value of $2l+1$ runs through from the odd numbers $m-k$ to $m-2$, the value of $2l+k+1-m$ runs through the even numbers from 0 to $k-2$, and the value of $2l+2k+1-m$ runs through the even numbers from k to $2k-2$.

Therefore, we have

$$\begin{aligned}
t_i^{(l)} &= \begin{cases} t_i^{(l-1)} + a'_{m+2l+1}, & i = 2l+1, 2l+k+1-m, 2l+2k+1-m; \\ t_i^{(l-1)}, & \text{otherwise.} \end{cases} \\
&= \begin{cases} t_i^{(l-1)} + a'_{m+i}, & i = 2l+1; \\ t_i^{(l-1)} + a'_{2m-k+i}, & i = 2l+k+1-m; \\ t_i^{(l-1)} + a'_{2m-2k+i}, & i = 2l+2k+1-m; \\ t_i^{(l-1)}, & \text{otherwise.} \end{cases} \\
&= \begin{cases} a'_{m+i}, & i = 1, 3, \dots, k-1; \\ a'_{m+i} + a'_{m-k+i}, & i = k+1, k+3, \dots, 2l+1; \\ a'_{m-k+i}, & i = 2l+3, 2l+5, \dots, m-2; \\ a'_i + a'_{2m-k+i}, & i = 0, 2, \dots, 2l+k+1-m; \\ a'_i + a'_{2m-2k+i}, & i = k, k+2, \dots, 2l+2k+1-m; \\ a'_i, & \text{otherwise.} \end{cases}
\end{aligned}$$

Then we can solve the final values for this case:

$$c_i = t_i^{(\frac{m-1}{2}-1)} = \begin{cases} a'_{m+i}, & i = 1, 3, \dots, k-1; \\ a'_{m+i} + a'_{m-k+i}, & i = k+1, k+3, \dots, m-2; \\ a'_i + a'_{2m-k+i}, & i = 0, 2, \dots, k-2; \\ a'_i + a'_{2m-2k+i}, & i = k, k+2, \dots, 2k-2; \\ a'_i, & i = 2k, 2k+2, \dots, m-1. \end{cases} \quad (6)$$

From the above equation we conclude that the total cost for computing squaring operation for this case is $\frac{m+k-1}{2}$ bit addition. The longest time delay to compute a c_i is the time taking to finish one bit addition.

2. If $2k = m - 1$.

In this case we have $2k - 1 = m - 2$. Thus from (5) it follows

$$t_i^{(\frac{k}{2}-1)} = \begin{cases} a'_{m+i}, & i = 1, 3, \dots, k-1; \\ a'_{m-k+i}, & i = k+1, k+3, \dots, m-2; \\ a'_i, & i = 0, 2, \dots, m-1. \end{cases}$$

Then for $\frac{k}{2} \leq l \leq \frac{m-1}{2} - 1$, we have

$$\sum_{i=0}^{m-1} t_i^{(l)} x^i = \sum_{i=0}^{m-1} t_i^{(l-1)} x^i + a'_{m+2l+1} x^{m+2l+1} \bmod f(x)$$

$$\begin{aligned}
&= \sum_{i=0}^{m-1} t_i^{(l-1)} x^i + a'_{m+2l+1} x^{2l+1} + a'_{m+2l+1} x^{2l+k+1} \bmod f(x) \\
&= \sum_{i=0}^{m-1} t_i^{(l-1)} x^i a'_{m+2l+1} x^{2l+1} + a'_{m+2l+1} x^{2l+k+1-m} \\
&\quad + a'_{m+2l+1} x^{2l+2k+1-m} \bmod f(x)
\end{aligned}$$

When l runs through from $\frac{k}{2}$ to $\frac{m-1}{2} - 1$, the value of $2l+1$ runs through from the odd numbers $k+1$ to $m-2$, the value of $2l+k+1-m$ runs through the even numbers from 0 to $k-2$, and the value of $2l+2k+1-m$ runs through the even numbers from k to $2k-2$.

Therefore, we have

$$\begin{aligned}
t_i^{(l)} &= \begin{cases} t_i^{(l-1)} + a'_{m+2l+1}, & i = 2l+1, 2l+k+1-m, 2l+2k+1-m; \\ t_i^{(l-1)}, & \text{otherwise.} \end{cases} \\
&= \begin{cases} t_i^{(l-1)} + a'_{m+i}, & i = 2l+1; \\ t_i^{(l-1)} + a'_{2m-k+i}, & i = 2l+k+1-m; \\ t_i^{(l-1)} + a'_{2m-2k+i}, & i = 2l+2k+1-m; \\ t_i^{(l-1)}, & \text{otherwise.} \end{cases} \\
&= \begin{cases} a'_{m+i}, & i = 1, 3, \dots, k-1; \\ a'_{m+i} + a'_{m-k+i}, & i = k+1, k+3, \dots, 2l+1; \\ a'_{m-k+i}, & i = 2l+3, 2l+5, \dots, m-2; \\ a'_i + a'_{2m-k+i}, & i = 0, 2, \dots, 2l+k+1-m; \\ a'_i + a'_{2m-2k+i}, & i = k, k+2, \dots, 2l+2k+1-m; \\ a'_i, & \text{otherwise.} \end{cases}
\end{aligned}$$

Then we can solve the final values for this case:

$$c_i = t_i^{(\frac{m-1}{2}-1)} = \begin{cases} a'_{m+i}, & i = 1, 3, \dots, k-1; \\ a'_{m+i} + a'_{m-k+i}, & i = k+1, k+3, \dots, m-2; \\ a'_i + a'_{2m-k+i}, & i = 0, 2, \dots, k-2; \\ a'_i + a'_{2m-2k+i}, & i = k, k+2, \dots, 2k-2; \\ a'_i, & i = 2k, 2k+2, \dots, m-1. \end{cases} \quad (7)$$

From the above equation it is clear that the total cost for computing squaring operation for this case is also $\frac{m+k-1}{2}$ bit addition.

3.3 m Is Even and $1 \leq k \leq \frac{m}{2}$ Is Odd

When the field is generated with an irreducible trinomial of form $f(x) = x^m + x^k + 1$, where m is even and $k \leq \frac{m}{2}$ is odd, similar analysis can be applied. In this case the complexity for a PB squaring operation in \mathbb{F}_{2^m} is $\frac{m+k-1}{2}$ bit additions if $k < \frac{m}{2}$, and $\frac{k+1}{2}$ bit additions if $k = \frac{m}{2}$ [6].

We summarize the results obtained from the three cases in this section in the following theorem:

Theorem 1. If there is an irreducible polynomial $f(x) = x^m + x^k + 1$, $1 \leq k \leq \frac{m}{2}$ over \mathbb{F}_2 , then a squaring operation in \mathbb{F}_{2^m} can be performed in

- (i) $\frac{m-1}{2}$ bit additions, if both m and k are odd.
- (ii) $\frac{m+k-1}{2}$ bit additions, if $m+k$ is odd and $k \neq \frac{m}{2}$.
- (iii) $\frac{k+1}{2}$ bit additions, if $k = \frac{m}{2}$.

4 Bit-Parallel Implementation

In hardware implementation, a bit addition in \mathbb{F}_2 can be realized using an XOR gate. If we denote the time propagation delay of an XOR gate by T_X , then the time delay of a hardware architecture can be measured in terms of gate delays.

For example, from (3a-3d) it can be seen that the most bit operations taken to compute a c_i are when $i = 1, 3, \dots, k-2$, as it is shown in (3d). Thus in this case the longest time propagation delay in a bit-parallel architecture for squaring is $2T_X$. The time delay for the other cases can be obtained from (2), (6), and (7) in a similar way.

The results on the complexity for a bit-parallel implementation of squaring operation are summarized as follows:

Theorem 2. If there is an irreducible polynomial $f(x) = x^m + x^k + 1$, $1 \leq k \leq \frac{m}{2}$ over \mathbb{F}_2 , then a bit-parallel hardware implementation of squaring operation in \mathbb{F}_{2^m} can be constructed with

- (i) $\frac{m-1}{2}$ XOR gates and the incurred time delay is $2T_X$, if both m and $k > 1$ are odd;
- (ii) $\frac{m-1}{2}$ XOR gates and the incurred time delay is T_X , if m is odd and $k = 1$;
- (iii) $\frac{m+k-1}{2}$ XOR gates and the incurred time delay is T_X , if m is odd and k is even;
- (iv) $\frac{m+k-1}{2}$ XOR gates and the incurred time delay is $2T_X$, if m is even and $1 < k < \frac{m}{2}$ is odd.
- (v) $\frac{m}{2}$ XOR gates and the incurred time delay is T_X , if m is even and $k = 1$.
- (vi) $\frac{k+1}{2}$ XOR gates and the incurred time delay is T_X , if m is even and $k = \frac{m}{2}$.

5 Concluding Remarks

Squaring operation is frequently required in elliptic curve cryptographic systems when an inversion or a point multiple operation is performed. Normal basis has been widely used because squaring operation using normal basis is only a cyclic shift of the coefficients. However, normal basis multiplication can be performed efficiently only when there is an optimal normal basis [5]. The results in this paper have shown that the complexity of a PB squaring operation is

very low, comparing to that of a multiplication operation ($O(m^2)$). This fact suggests that polynomial basis might be a good replacement for normal basis in many cryptographic application, since the prevailing existence of irreducible trinomial [1], comparing to that of optimal normal basis.

References

1. Blake, I.F., Gao, S., Lambert, R.: Constructive Problems for Irreducible Polynomials over Finite Fields. Canadian Workshop on IT, Springer-Verlag, 1993
2. Hasan, M. A., Wang, M., Bhargava, V. K.: A modified Massey-Omura parallel multiplier for a class of finite fields. IEEE Trans. Comput. **42** (1993) 1278-1280
3. Sunar, B., Koc, C. K.: Mastrovito Multiplier for all trinomials. IEEE Trans. Comput. **48** (1999) 522-527
4. Menezes, A.: Private communication. March, 2000.
5. Mullin, R., Onyszchuk, I., Vanstone, S.A., Wilson, R.: Optimal normal bases in $GF(p^n)$. Disc. Appl. Math. **22** (1988) 149-161
6. Wu, H.: Efficient Computations in Finite Fields with Cryptographic Significance. Ph.D Thesis, University of Waterloo, Waterloo, Canada, 1998

Dynamic Multi-threshold Metering Schemes

Carlo Blundo¹, Annalisa De Bonis¹, Barbara Masucci¹, and
Douglas R. Stinson²

¹ Dipartimento di Informatica ed Applicazioni, Università di Salerno,
84081 Baronissi (SA), Italy,
`{carblu,debonis,masucci}@dia.unisa.it`
`http://www.dia.unisa.it/~{carblu,masucci}`

² Department of Combinatorics and Optimization, University of Waterloo,
Waterloo, Ontario, N2L 3G1, Canada,
`dstinson@cacr.math.uwaterloo.ca`
`http://www.cacr.math.uwaterloo.ca/~dstinson`

Abstract. A *metering scheme* is a protocol in which an audit agency is able to measure the interaction between clients and servers on the web during a certain number of time frames. Naor and Pinkas [7] considered metering schemes in which any server is able to construct a proof to be sent to the audit agency if and only if it has been visited by at least a number, say h , of clients in a given time frame. In their schemes the parameter h is fixed and is the same for any server and any time frame. In this paper we introduce *dynamic multi-threshold metering schemes*, that are metering schemes in which there is a threshold associated to any server for any time frame. We mainly focus on the efficiency of dynamic multi-threshold metering schemes, by minimizing the information received and distributed by clients. This is important because the clients participating in the metering process do not receive any money from the audit agency.

Keywords: Metering Schemes, Security, Cryptography, Entropy.

1 Introduction

Most of the revenues of web sites come from advertisement payments. Web advertisers must have a way to measure the exposure of their ads by obtaining usage statistics about web sites which contain their ads. Indeed, the amount of money charged to display ads depends on the number of visits received by the web site. Consequently, advertisers should prevent the web sites from inflating the count of their visits in order to demand more money. Hence, there should be a mechanism which ensures the validity and accuracy of usage measurements against fraud attempts by servers (web sites) and clients (visitors). In a typical scenario there are many servers and clients, and an audit agency whose task is to measure the interaction between the servers and the clients. A system for measuring the amount of services performed by the servers is called *metering scheme*.

Naor and Pinkas [7] proposed metering schemes in which any server is able to present to the audit agency a short proof for the number of client visits it has received in a given time frame. In their schemes all servers are associated to a threshold h , and are able to compute their proofs for a certain time frame if and only if they have been visited by a number of clients larger than or equal to h in that time frame. The schemes proposed by Naor and Pinkas are also efficient: the task for the audit agency in sending information to clients and servers is very simple, as well as the task for the servers in computing their proofs. Recently, different kinds of metering schemes have been proposed. Metering schemes for ramp structures [1,3] have been introduced in order to reduce the overhead to the overall communication due to the metering process. Metering schemes with pricing [1,5] have been introduced in order to have a more flexible payment system. Finally, metering schemes for general access structures [6] have been introduced in order to measure the interaction between servers and particular groups of clients.

In metering schemes considered by Naor and Pinkas [7] the parameter h is fixed and is the same for any server and any time frame. This is acceptable whenever there is a long-term relationship between the audit agency and the servers. In order to measure any number of visits in any granularity we introduce *dynamic multi-threshold metering schemes*, which are metering schemes in which there is a threshold h_j^t associated to any server \mathcal{S}_j for any time frame t .

Dynamic multi-threshold metering schemes involve distributing information to clients and servers. Obviously, such information distribution affects the overall communication complexity. Therefore, it is important to construct schemes whose overhead to the overall communication is as small as possible. We mainly focus on the efficiency of dynamic multi-threshold metering schemes, by minimizing the information received and distributed by clients. This is important because the clients participating in the metering process do not receive any money from the audit agency. In this paper we provide lower bounds on the size of the information received and distributed by clients and we present a scheme achieving these lower bounds.

2 The Model

Consider the following scenario: there are n clients, m servers and an audit agency A which is interested in counting the client visits to the servers in τ different time frames. For any $i = 1, \dots, n$ and $j = 1, \dots, m$, we denote by \mathcal{C}_i the i -th client and by \mathcal{S}_j the j -th server.

There is an *initialization phase* in which the audit agency A distributes some information to any client over a private channel. For any $i = 1, \dots, n$, we denote by c_i the information that the audit agency A gives to the client \mathcal{C}_i . Moreover, we denote by C_i the set of all values that c_i can assume. Given a set of client indices $Z = \{1, \dots, \alpha\} \subseteq \{1, \dots, n\}$, we denote by C_Z the cartesian product $C_1 \times \dots \times C_\alpha$.

At the *beginning of any time frame* the audit agency A distributes to any server a piece of information which depends on the identity of the server and on the time frame. For any $j = 1, \dots, m$ and $t = 1, \dots, \tau$, we denote by s_j^t the information that the audit agency A gives to the server \mathcal{S}_j at the beginning of time frame t . Moreover, we denote by S_j^t the set of all values that s_j^t can assume. Given a set of server indices $B = \{1, \dots, \beta\} \subseteq \{1, \dots, m\}$, we denote by S_B^t the cartesian product $S_1^t \times \dots \times S_\beta^t$.

A *regular operation* consists in a client visit to a server during a time frame. During such a visit the client gives to the visited server a piece of information which depends on its private information, on the identity of the server, and on the time frame during which the client visits the server. For any $i = 1, \dots, n$, $j = 1, \dots, m$, and $t = 1, \dots, \tau$, we denote by $c_{i,j}^t$ the information that the client \mathcal{C}_i sends to the server \mathcal{S}_j when visiting it in time frame t . Moreover, we denote by $C_{i,j}^t$ the set of all values that $c_{i,j}^t$ can assume. Given a set of server indices $B = \{1, \dots, \beta\} \subseteq \{1, \dots, m\}$, we denote by $C_{i,B}^t$ the cartesian product $C_{i,1}^t \times \dots \times C_{i,\beta}^t$. Moreover, given a set of client indices $Z = \{1, \dots, \alpha\} \subseteq \{1, \dots, n\}$, we denote by $C_{Z,B}^t$ the cartesian product $C_{1,B}^t \times \dots \times C_{\alpha,B}^t$. For any $j = 1, \dots, m$ and $t = 1, \dots, \tau$, we denote by $X_{j,(d_j)}^t$ the set of the d_j client visits received by server \mathcal{S}_j in time frame t .

During the *proof computation stage* any server \mathcal{S}_j which has received at least h_j^t visits during time frame t is able to compute its proof for time frame t , as function of the information provided by the h_j^t clients and the information s_j^t provided by the audit agency A at the beginning of the time frame t . For any $j = 1, \dots, m$ and $t = 1, \dots, \tau$, we denote by p_j^t the proof computed by the server \mathcal{S}_j when it has been visited by at least h_j^t distinct clients in time frame t . Moreover, we denote by P_j^t the set of all values that p_j^t can assume. Given a set of server indices $B = \{1, \dots, \beta\} \subseteq \{1, \dots, m\}$, we denote by P_B^t the cartesian product $P_1^t \times \dots \times P_\beta^t$.

During the *proof verification stage* the audit agency A verifies the proofs received by servers and decides on the amount of money to be paid to servers. If the proof received from a server at the end of a time frame is correct, then A pays the server for its services.

A *corrupt server* can be assisted by corrupt clients and other corrupt servers in order to inflate the count of its visits. A corrupt client \mathcal{C}_i can donate to a corrupt server the whole private information received by the audit agency during the initialization phase. We assume that the number of corrupt clients is c , where $1 \leq c < \min_{j=1,\dots,m} \min_{t=1,\dots,\tau} h_j^t$. A corrupt server can donate to another corrupt server the private information received from the audit agency at the beginning of any time frame in addition to the information received from clients in previous time frames and in the actual time frame. For any $i = 1, \dots, n$ and $t = 1, \dots, \tau$, we denote by $V_j^{[t]}$ all the information received by a corrupt server \mathcal{S}_j in time frames $1, \dots, t$. This information includes the sets of client visits received by server \mathcal{S}_j in time frames $1, \dots, t$. We also define $V_j^{[0]} = \emptyset$, for any corrupt server \mathcal{S}_j . We assume that the maximum number of corrupt servers

is s , where $1 \leq s \leq m$. For the reader's convenience, the notations used in this section are summarized in Appendix B.

In this paper with a boldface capital letter, say \mathbf{X} , we denote a random variable taking value on a set denoted by the corresponding capital letter X according to some probability distribution $\{Pr_{\mathbf{X}}(x)\}_{x \in X}$. The values such a random variable can take are denoted by the corresponding lower letter. Given a random variable \mathbf{X} we denote with $H(\mathbf{X})$ the Shannon entropy of $\{Pr_{\mathbf{X}}(x)\}_{x \in X}$ (for some basic properties of entropy, consult the Appendix A).

We formally define dynamic multi-threshold metering schemes by using the entropy approach, as done in [1,3,5,6]. We use the entropy approach mainly because this leads to a compact and simple description of the schemes and because the entropy approach takes into account all probability distributions on the sets of the proofs computed by the servers.

Definition 1. An $(n, m, \tau, c, s, \{h_j\}_{j=1, \dots, m}^{t=1, \dots, \tau})$ dynamic multi-threshold metering scheme is a protocol to measure the interaction between n clients and m servers during τ time frames in such a way that the following properties are satisfied:

1. Any client is able to compute the information needed to visit any server in any time frame:
Formally, it holds that $H(\mathbf{C}_{i,j}^t | \mathbf{C}_i) = 0$ for $i = 1, \dots, n$, $j = 1, \dots, m$, and $t = 1, \dots, \tau$.
2. Any server \mathcal{S}_j which has received h_j^t client visits during time frame t and the message provided by A at the beginning of the time frame t can compute its proof for t :
Formally, it holds that $H(\mathbf{P}_j^t | \mathbf{X}_{j,(h_j^t)}^t \mathbf{S}_j^t) = 0$, for $j = 1, \dots, m$ and $t = 1, \dots, \tau$.
3. Let us consider a coalition of α corrupt clients $\mathcal{C}_1, \dots, \mathcal{C}_\alpha$ and β corrupt servers $\mathcal{S}_1, \dots, \mathcal{S}_\beta$, where $0 \leq \alpha \leq c < \min_{j=1, \dots, m} \min_{t=1, \dots, \tau} h_j^t$ and $1 \leq \beta \leq s$, and let $B = \{1, \dots, \beta\}$. Assume that at some time frame t each server \mathcal{S}_j in the coalition has been visited by less than $h_j^t - \alpha$ clients and has received the information by A . Then, the servers in the coalition have no information on their proofs for t :
Formally, it holds that $H(\mathbf{P}_B^t | \mathbf{C}_1 \dots \mathbf{C}_\alpha \mathbf{X}_{1,(d_1)}^t \dots \mathbf{X}_{\beta,(d_\beta)}^t \mathbf{S}_B^t \mathbf{V}_B^{[t-1]}) = H(\mathbf{P}_B^t)$, where $d_j < h_j^t - \alpha$, for $j = 1, \dots, \beta$.

Notice that Naor and Pinkas [7] considered metering schemes which are “static” and with “single threshold”, i.e., where $h_j^t = h$ for $j = 1, \dots, m$ and $t = 1, \dots, \tau$. Moreover, their schemes do not require communication between audit agency and servers at the beginning of any time frame.

3 A Dynamic Multi-threshold Metering Protocol

In this section we present a dynamic multi-threshold metering scheme which is optimal with respect to the bounds (4) and (5) presented in Section 4. The protocol is a generalization of Naor and Pinkas metering scheme [7].

Initialization: For $j = 1, \dots, m$ and $t = 1, \dots, \tau$, let h_j^t be the threshold associated to the server \mathcal{S}_j in time frame t and let $h = \max_{j=1, \dots, m} \max_{t=1, \dots, \tau} h_j^t + 1$. The audit agency A chooses a random polynomial $Q(x, y)$ of degree $h - 1$ in x and $s\tau - 1$ in y over $GF(q)$, where q is a sufficiently large prime number. Afterwards, A sends the univariate polynomial $Q(i, y)$, which is of degree $s\tau - 1$, to each client \mathcal{C}_i .

Beginning of a Time Frame: At the beginning of time frame t , for any server \mathcal{S}_j , the audit agency A evaluates the polynomial $Q(x, j \circ t)$ in $h - h_j^t$ points other than $1, \dots, n$ and sends these values to \mathcal{S}_j . The argument $j \circ t$ denotes the concatenation of j and t , and we assume for simplicity that $j \circ t$ is in $GF(q)$ and that no distinct two pairs (j, t) and (j', t') are mapped to the same element.

Regular Operation: When the client \mathcal{C}_i visits the server \mathcal{S}_j in time frame t , it sends the value $Q(i, j \circ t)$ to \mathcal{S}_j .

Proof Generation and Verification: Assume that the server \mathcal{S}_j has been visited by at least h_j^t different clients in time frame t . Then, knowing the $h - h_j^t$ points of $Q(x, j \circ t)$ provided by the audit agency at the beginning of time frame t , the server can perform a Lagrange interpolation and reconstruct the polynomial $Q(x, j \circ t)$. Then, it can compute the value $Q(0, j \circ t)$, which constitutes the proof that the server sends to the audit agency. The audit agency can easily verify this value.

3.1 Security of the Scheme

In this section we prove that the scheme presented in Section 3 satisfies Properties 1, 2, and 3 of Definition 1.

It is immediate to verify that the scheme satisfies Property 1 of Definition 1. Indeed, for any $i = 1, \dots, n$, the information given by the audit agency to the client \mathcal{C}_i consists of the univariate polynomial $Q(i, y)$ and for any $j = 1, \dots, m$ and $t = 1, \dots, \tau$, the information given to the server \mathcal{S}_j by client \mathcal{C}_i in time frame t is obtained by evaluating the univariate polynomial $Q(i, y)$ at $j \circ t$.

It is also easy to verify that the scheme satisfies Property 2 of Definition 1. Assume that a server \mathcal{S}_j has been visited by h_j^t clients in time frame t and that it has received $h - h_j^t$ points of $Q(x, j \circ t)$ from the audit agency at the beginning of time frame t . Therefore, the server \mathcal{S}_j knows h points of the polynomial $Q(x, j \circ t)$ and can perform a Lagrange interpolation on it. Afterwards, it can compute its proof $Q(0, j \circ t)$ by evaluating the polynomial $Q(x, j \circ t)$ at the point 0.

Finally, we prove that the scheme satisfies Property 3 of Definition 1. We consider the worst possible case in which c corrupt clients decide to cooperate with s corrupt servers at time frame τ . Moreover we assume that the corrupt servers have collected the maximum possible information during the previous time frames $1, \dots, \tau - 1$. In other words, we assume that each corrupt client \mathcal{C}_i

gives its polynomial $Q(i, y)$ to all servers in the coalition, and that any corrupt server \mathcal{S}_j in the coalition knows the polynomial $Q(x, j \circ t)$ for $t = 1, \dots, \tau - 1$.

In order to compute its proof $Q(0, j \circ \tau)$ for time frame τ , any server \mathcal{S}_j should be able to interpolate either the polynomial $Q(x, j \circ \tau)$ or the bivariate polynomial $Q(x, y)$. Notice that for any $j, k \in \{1, \dots, s\}$, with $j \neq k$, the information held by the server \mathcal{S}_k is of no help in computing the polynomial $Q(x, j \circ \tau)$. Assume $g_j = h_j^\tau - c - 1$ be the number of client visits received by server \mathcal{S}_j during time frame τ . Each corrupt client \mathcal{C}_i donates to \mathcal{S}_j the polynomial $Q(i, y)$ from which \mathcal{S}_j can compute the value $Q(i, j \circ \tau)$. Since there are c corrupt clients, \mathcal{S}_j can compute c values of $Q(x, j \circ \tau)$ in addition to those provided by the g_j visits performed by non corrupt clients. Since the server \mathcal{S}_j has also received $h - h_j^\tau$ points of $Q(x, j \circ \tau)$ by the audit agency at the beginning of time frame τ , the overall number of points of $Q(x, j \circ \tau)$ known to \mathcal{S}_j is $g_j + c + h - h_j^\tau = h - 1$. Therefore, the server obtains a linear system of $h - 1$ equations in h unknowns. For any choice of a value in $GF(q)$, there is a polynomial $R(x, j \circ \tau)$ which is consistent with the information held by the server. Since there are q such polynomials, the probability of the server in guessing its proof for time frame τ is at most $1/q$.

Alternatively, the coalition of corrupt servers might try to interpolate the polynomial $Q(x, y)$ in order to compute the proofs. The information that a corrupt client \mathcal{C}_i gives to a corrupt server is equivalent to the $s\tau$ coefficients of its polynomial $Q(i, y)$. For $j = 1, \dots, s$, the information collected by each corrupt server \mathcal{S}_j at the beginning of time frame τ is constituted by the information provided by the audit agency at the beginning of any time frame $t = 1, \dots, \tau$, which consists in $h - h_j^t$ coefficients of $Q(x, j \circ t)$, in addition to the information provided by clients during each time frame $t = 1, \dots, \tau - 1$, which consists in h_j^t coefficients of $Q(x, j \circ t)$. Hence, at the beginning of time frame τ each corrupt server holds $(\tau - 1)h$ coefficients of $Q(x, y)$ and $h - h_j^\tau$ coefficients of $Q(x, j \circ \tau)$. Suppose that in time frame τ each server \mathcal{S}_j , $j \in \{1, \dots, s\}$, receives $g_j \leq h_j^\tau - \alpha - 1$ regular visits from clients. Then, the overall information on $Q(x, y)$ held by the coalition of corrupt servers and clients at the end of time frame τ consists of

$$cs\tau + s(\tau - 1)h + \sum_{j=1}^s (h - h_j^\tau) + \sum_{j=1}^s g_j - cs(\tau - 1) \quad (1)$$

points. The first term of (1) corresponds to the information donated by the c corrupt clients, the second term corresponds to the information collected by the s corrupt servers during time frames $1, \dots, \tau - 1$, the third term corresponds to the information provided by the audit agency at the beginning of time frame τ , the fourth term corresponds to the information provided by client visits at time frame τ , and the last term corresponds to the information which has been counted twice. Since $g_j \leq h_j^\tau - \alpha - 1$ for $j = 1, \dots, s$, it is easy to see that expression (1) is less than or equal to $hs\tau - s$. Therefore, the servers obtain a system of at most $hs\tau - s$ equations in $hs\tau$ unknowns. For any choice of s values in $GF(q)$, there is a polynomial $R(x, y)$ which is consistent with the information

held by the servers in the coalition. Since there are q^s such polynomials, then the corrupt servers $\mathcal{S}_1, \dots, \mathcal{S}_s$ have probability at most $1/q^s$ of guessing their proofs for time frame τ .

4 Lower Bounds on the Size of the Information Distributed to Clients and Servers

Dynamic multi-threshold metering schemes involve distributing information to clients and servers. In this section we provide lower bounds on the size of the information received by clients from the audit agency and distributed by clients to servers in dynamic multi-threshold metering schemes.

In order to prove our results we will resort to the two following technical lemmas.

Lemma 2. *Let \mathbf{A} and \mathbf{E} be two random variables such that $H(\mathbf{A}|\mathbf{E}) = 0$. Then, for any two random variables \mathbf{F} and \mathbf{G} , it holds that*

$$H(\mathbf{G}|\mathbf{AEF}) = H(\mathbf{G}|\mathbf{EF}).$$

Proof. Consider the mutual information $I(\mathbf{A}; \mathbf{G}|\mathbf{EF})$. From (12) of Appendix A it holds that

$$H(\mathbf{A}|\mathbf{EF}) - H(\mathbf{A}|\mathbf{EFG}) = H(\mathbf{G}|\mathbf{EF}) - H(\mathbf{G}|\mathbf{AEF}).$$

From (13) of Appendix A we have that $H(\mathbf{A}|\mathbf{EFG}) \leq H(\mathbf{A}|\mathbf{EF}) \leq H(\mathbf{A}|\mathbf{E})$. Since $H(\mathbf{A}|\mathbf{E}) = 0$, it follows that

$$H(\mathbf{G}|\mathbf{AEF}) = H(\mathbf{G}|\mathbf{EF}).$$

□

Lemma 3. *Let \mathbf{E} , \mathbf{F} , and \mathbf{G} be three random variables such that $H(\mathbf{G}|\mathbf{EF}) = 0$ and $H(\mathbf{G}|\mathbf{E}) = H(\mathbf{G})$. Then, it holds that*

$$H(\mathbf{F}|\mathbf{E}) = H(\mathbf{G}) + H(\mathbf{F}|\mathbf{EG}).$$

Proof. Consider the mutual information $I(\mathbf{F}; \mathbf{G}|\mathbf{E})$. From (12) of Appendix A it holds that

$$H(\mathbf{F}|\mathbf{E}) - H(\mathbf{F}|\mathbf{EG}) = H(\mathbf{G}|\mathbf{E}) - H(\mathbf{G}|\mathbf{EF}).$$

Since $H(\mathbf{G}|\mathbf{EF}) = 0$ and $H(\mathbf{G}|\mathbf{E}) = H(\mathbf{G})$, then it follows that $H(\mathbf{F}|\mathbf{E}) = H(\mathbf{G}) + H(\mathbf{F}|\mathbf{EG})$. □

The next lemma immediately follows from Definition 1. Recall that for any sets of client and server indices $Z = \{1, \dots, \alpha\} \subseteq \{1, \dots, n\}$ and $B = \{1, \dots, \beta\} \subseteq \{1, \dots, m\}$, respectively, we denote by $c_{Z,B}^t$ the information given by clients $\mathcal{C}_1, \dots, \mathcal{C}_\alpha$ to servers $\mathcal{S}_1, \dots, \mathcal{S}_\beta$ during their visits in time frame t .

Lemma 4. Let \mathcal{M} be an $(n, m, \tau, c, s, \{h_j\}_{j=1, \dots, m}^{t=1, \dots, \tau})$ dynamic multi-threshold metering scheme. Let $Z = \{1, \dots, \alpha\}$ be a set of client indices and let $B = \{1, \dots, \beta\}$ be a set of server indices. Then, for any time frame $t = 1, \dots, \tau$, it holds that

$$H(\mathbf{C}_{Z,B}^t | \mathbf{C}_Z) = 0.$$

Proof. We have that

$$\begin{aligned} H(\mathbf{C}_{Z,B}^t | \mathbf{C}_Z) &\leq \sum_{i=1}^{\alpha} \sum_{j=1}^{\beta} H(\mathbf{C}_{i,j}^t | \mathbf{C}_Z) \text{ (from (15) of Appendix A)} \\ &\leq \sum_{i=1}^{\alpha} \sum_{j=1}^{\beta} H(\mathbf{C}_{i,j}^t | \mathbf{C}_i) \text{ (from (13) of Appendix A)} \\ &= 0 \text{ (from Property 1 of Definition 1).} \end{aligned}$$

□

The next lemma will be a useful tool to prove a lower bound on the size of the information distributed to servers from clients during their visits.

Lemma 5. Let \mathcal{M} be an $(n, m, \tau, c, s, \{h_j\}_{j=1, \dots, m}^{t=1, \dots, \tau})$ dynamic multi-threshold metering scheme. Let $\mathcal{S}_1, \dots, \mathcal{S}_\beta$ be a coalition of $\beta \leq s$ corrupt servers and let $B = \{1, \dots, \beta\}$. Let \mathcal{C}_i be a client and for $j = 1, \dots, \beta$ and $t = 1, \dots, \tau$, let $X_{j, (h_j^t - 1)}^t$ be a set of visits from $h_j^t - 1$ clients other than \mathcal{C}_i to server \mathcal{S}_j in time frame t . Then, for any $t = 1, \dots, \tau$ and $i = 1, \dots, n$, it holds that

$$H(\mathbf{C}_{i,B}^t | \mathbf{X}_{1, (h_1^t - 1)}^t \dots \mathbf{X}_{\beta, (h_\beta^t - 1)}^t \mathbf{S}_B^t \mathbf{V}_B^{[t-1]}) \geq H(\mathbf{P}_B^t).$$

Proof. Let $\mathcal{C}_1, \dots, \mathcal{C}_\alpha$ be a coalition of $\alpha \leq c$ corrupt servers other than \mathcal{C}_i . Let us consider the random variables $\mathbf{E} = \mathbf{C}_1 \dots \mathbf{C}_\alpha \mathbf{X}_{1, (h_1^t - \alpha - 1)}^t \dots \mathbf{X}_{\beta, (h_\beta^t - \alpha - 1)}^t \mathbf{S}_B^t \mathbf{V}_B^{[t-1]}$, $\mathbf{A} = \mathbf{C}_{1,B}^t \dots \mathbf{C}_{\alpha,B}^t$, $\mathbf{F} = \mathbf{C}_{i,B}^t$, and $\mathbf{G} = \mathbf{P}_B^t$. We have that

$$\begin{aligned} H(\mathbf{C}_{1,B}^t \dots \mathbf{C}_{\alpha,B}^t | \mathbf{C}_1 \dots \mathbf{C}_\alpha \mathbf{C}_{i,B}^t \mathbf{X}_{1, (h_1^t - \alpha - 1)}^t \dots \mathbf{X}_{\beta, (h_\beta^t - \alpha - 1)}^t \mathbf{S}_B^t \mathbf{V}_B^{[t-1]}) \\ \leq H(\mathbf{C}_{1,B}^t \dots \mathbf{C}_{\alpha,B}^t | \mathbf{C}_1 \dots \mathbf{C}_\alpha) \text{ (from (13) of Appendix A)} \\ = 0 \text{ (from Lemma 4).} \end{aligned}$$

Hence, \mathbf{A} , \mathbf{E} , and \mathbf{F} verify the hypothesis of Lemma 2, and one has $H(\mathbf{G} | \mathbf{EF}) = H(\mathbf{G} | \mathbf{AEF})$, that is,

$$\begin{aligned} H(\mathbf{P}_B^t | \mathbf{C}_1 \dots \mathbf{C}_\alpha \mathbf{C}_{i,B}^t \mathbf{X}_{1, (h_1^t - \alpha - 1)}^t \dots \mathbf{X}_{\beta, (h_\beta^t - \alpha - 1)}^t \mathbf{S}_B^t \mathbf{V}_B^{[t-1]}) \\ = H(\mathbf{P}_B^t | \mathbf{C}_{1,B}^t \dots \mathbf{C}_{\alpha,B}^t \mathbf{C}_1 \dots \mathbf{C}_\alpha \mathbf{C}_{i,B}^t \mathbf{X}_{1, (h_1^t - \alpha - 1)}^t \dots \mathbf{X}_{\beta, (h_\beta^t - \alpha - 1)}^t \mathbf{S}_B^t \mathbf{V}_B^{[t-1]}) \\ \leq H(\mathbf{P}_B^t | \mathbf{C}_{1,B}^t \dots \mathbf{C}_{\alpha,B}^t \mathbf{C}_{i,B}^t \mathbf{X}_{1, (h_1^t - \alpha - 1)}^t \dots \mathbf{X}_{\beta, (h_\beta^t - \alpha - 1)}^t \mathbf{S}_B^t \mathbf{V}_B^{[t-1]}) \\ \text{(from (13) of Appendix A)} \\ = H(\mathbf{P}_B^t | \mathbf{X}_{1, (h_1^t)}^t \dots \mathbf{X}_{\beta, (h_\beta^t)}^t \mathbf{S}_B^t \mathbf{V}_B^{[t-1]}) \end{aligned}$$

$$\begin{aligned}
&\leq \sum_{j=1}^{\beta} H(\mathbf{P}_j^t | \mathbf{X}_{j,(h_j^t)}^t \mathbf{S}_j^t) \text{ (from (13) and (15) of Appendix A)} \\
&= 0 \text{ (from Property 2 of Definition 1).}
\end{aligned}$$

From Property 3 of Definition 1 we have that

$$H(\mathbf{P}_B^t | \mathbf{C}_1 \dots \mathbf{C}_\alpha \mathbf{X}_{1,(h_1^t-\alpha-1)}^t \dots \mathbf{X}_{\beta,(h_\beta^t-\alpha-1)}^t \mathbf{S}_B^t \mathbf{V}_B^{[t-1]}) = H(\mathbf{P}_B^t).$$

Hence, $\mathbf{E} = \mathbf{C}_1 \dots \mathbf{C}_\alpha \mathbf{X}_{1,(h_1^t-\alpha-1)}^t \dots \mathbf{X}_{\beta,(h_\beta^t-\alpha-1)}^t \mathbf{S}_B^t \mathbf{V}_B^{[t-1]}$, $\mathbf{F} = \mathbf{C}_{i,B}^t$, and $\mathbf{G} = \mathbf{P}_B^t$ verify the hypothesis of Lemma 3 and one has $H(\mathbf{F}|\mathbf{E}) = H(\mathbf{G}) + H(\mathbf{F}|\mathbf{E}\mathbf{G})$, that is

$$\begin{aligned}
&H(\mathbf{C}_{i,B}^t | \mathbf{C}_1 \dots \mathbf{C}_\alpha \mathbf{X}_{1,(h_1^t-\alpha-1)}^t \dots \mathbf{X}_{\beta,(h_\beta^t-\alpha-1)}^t \mathbf{S}_B^t \mathbf{V}_B^{[t-1]}) \\
&= H(\mathbf{P}_B^t) + H(\mathbf{C}_{i,B}^t | \mathbf{C}_1 \dots \mathbf{C}_\alpha \mathbf{X}_{1,(h_1^t-\alpha-1)}^t \dots \mathbf{X}_{\beta,(h_\beta^t-\alpha-1)}^t \mathbf{S}_B^t \mathbf{V}_B^{[t-1]} \mathbf{P}_B^t) \\
&\geq H(\mathbf{P}_B^t) \text{ (from (7) of Appendix A).} \tag{2}
\end{aligned}$$

Moreover, $\mathbf{A} = \mathbf{C}_{1,B}^t \dots \mathbf{C}_{\alpha,B}^t$, $\mathbf{E} = \mathbf{C}_1 \dots \mathbf{C}_\alpha \mathbf{X}_{1,(h_1^t-\alpha-1)}^t \dots \mathbf{X}_{\beta,(h_\beta^t-\alpha-1)}^t \mathbf{S}_B^t \mathbf{V}_B^{[t-1]}$, and $\mathbf{F} = \mathbf{C}_{i,B}^t$ verify the hypothesis of Lemma 2 and one has $H(\mathbf{F}|\mathbf{E}) = H(\mathbf{F}|\mathbf{A}\mathbf{E})$, that is

$$\begin{aligned}
&H(\mathbf{C}_{i,B}^t | \mathbf{C}_1 \dots \mathbf{C}_\alpha \mathbf{X}_{1,(h_1^t-\alpha-1)}^t \dots \mathbf{X}_{\beta,(h_\beta^t-\alpha-1)}^t \mathbf{S}_B^t \mathbf{V}_B^{[t-1]}) \\
&= H(\mathbf{C}_{i,B}^t | \mathbf{C}_{1,B}^t \dots \mathbf{C}_{\alpha,B}^t \mathbf{C}_1 \dots \mathbf{C}_\alpha \mathbf{X}_{1,(h_1^t-\alpha-1)}^t \dots \mathbf{X}_{\beta,(h_\beta^t-\alpha-1)}^t \mathbf{S}_B^t \mathbf{V}_B^{[t-1]}) \\
&\leq H(\mathbf{C}_{i,B}^t | \mathbf{C}_{1,B}^t \dots \mathbf{C}_{\alpha,B}^t \mathbf{X}_{1,(h_1^t-\alpha-1)}^t \dots \mathbf{X}_{\beta,(h_\beta^t-\alpha-1)}^t \mathbf{S}_B^t \mathbf{V}_B^{[t-1]}) \\
&\quad \text{(from (13) of Appendix A)} \\
&= H(\mathbf{C}_{i,B}^t | \mathbf{X}_{1,(h_1^t-1)}^t \dots \mathbf{X}_{\beta,(h_\beta^t-1)}^t \mathbf{S}_B^t \mathbf{V}_B^{[t-1]}) \tag{3}
\end{aligned}$$

Therefore, the lemma follows from inequalities (3) and (2). \square

The next corollary immediately follows from Lemma 5. It implicitly shows that the size of the information each client has to give out when visiting a server is lower bounded by the size of the proof the server could reconstruct.

Corollary 6. *Let \mathcal{M} be an $(n, m, \tau, c, s, \{h_j\}_{j=1,\dots,m}^{t=1,\dots,\tau})$ dynamic multi-threshold metering scheme. For any $i = 1, \dots, n$, $j = 1, \dots, m$, and $t = 1, \dots, \tau$, it holds that*

$$H(\mathbf{C}_{i,j}^t) \geq H(\mathbf{P}_j^t).$$

If the proofs for the servers are uniformly chosen in a finite field F , that is $H(\mathbf{P}_j^t) = \log |F|$ for any $j = 1, \dots, m$ and $t = 1, \dots, \tau$, then from Corollary 6 and from (6) of Appendix A it holds that

$$\log |C_{i,j}^t| \geq \log |F| \tag{4}$$

for $i = 1, \dots, n$, $j = 1, \dots, m$, and $t = 1, \dots, \tau$. This bound is tight, as in Section 3 we have presented a protocol for an $(n, m, \tau, c, s, \{h_j\}_{j=1, \dots, m}^{t=1, \dots, \tau})$ dynamic multi-threshold metering scheme in which the clients distribute *exactly* this information to servers during their visits.

In order to prove a lower bound on the size of the information distributed to clients we need the next lemma.

Lemma 7. *Let \mathcal{M} be an $(n, m, \tau, c, s, \{h_j\}_{j=1, \dots, m}^{t=1, \dots, \tau})$ dynamic multi-threshold metering scheme. Let $\mathcal{S}_1 \dots, \mathcal{S}_\beta$ be a coalition of $\beta \leq s$ corrupt servers and let $B = \{1, \dots, \beta\}$. Let $Z \subseteq \{1, \dots, n\}$ be a set of client indices. Then, it holds that*

$$H(\mathbf{C}_Z) \geq \sum_{t=1}^{\tau} H(\mathbf{C}_{Z,B}^t | \mathbf{V}_B^{[t-1]}).$$

Proof. We have that

$$\begin{aligned} H(\mathbf{C}_{Z,B}^1 \dots \mathbf{C}_{Z,B}^\tau | \mathbf{C}_Z) &\leq \sum_{t=1}^{\tau} H(\mathbf{C}_{Z,B}^t | \mathbf{C}_Z) \text{ (from (15) and (13) of Appendix A)} \\ &= 0 \text{ (from Lemma 4).} \end{aligned}$$

Therefore, applying Lemma 3 with $\mathbf{F} = \mathbf{C}_{Z,B}^1 \dots \mathbf{C}_{Z,B}^\tau$ and $\mathbf{D} = \mathbf{C}_Z$ we get

$$\begin{aligned} H(\mathbf{C}_Z) &= H(\mathbf{C}_{Z,B}^1 \dots \mathbf{C}_{Z,B}^\tau) + H(\mathbf{C}_Z | \mathbf{C}_{Z,B}^1 \dots \mathbf{C}_{Z,B}^\tau) \\ &\geq H(\mathbf{C}_{Z,B}^1 \dots \mathbf{C}_{Z,B}^\tau) \text{ (from (7) of Appendix A)} \\ &= H(\mathbf{C}_{Z,B}^1) + \sum_{t=2}^{\tau} H(\mathbf{C}_{Z,B}^t | \mathbf{C}_{Z,B}^1 \dots \mathbf{C}_{Z,B}^{t-1}) \text{ (from (14) of Appendix A)} \\ &\geq \sum_{t=1}^{\tau} H(\mathbf{C}_{Z,B}^t | \mathbf{V}_B^{[t-1]}). \end{aligned}$$

□

The next lemma provides a lower bound on the size of the information distributed to clients during the initialization phase in dynamic multi-threshold metering schemes. It states that the information that must be kept secret by clients grows linearly with the number of time frames and the size of the coalition of corrupt servers.

Lemma 8. *Let \mathcal{M} be an $(n, m, \tau, c, s, \{h_j\}_{j=1, \dots, m}^{t=1, \dots, \tau})$ dynamic multi-threshold metering scheme. Let $\mathcal{S}_1 \dots, \mathcal{S}_\beta$ be a coalition of $\beta \leq s$ corrupt servers and let $B = \{1, \dots, \beta\}$. For any $i = 1, \dots, n$, it holds that*

$$H(\mathbf{C}_i) \geq \sum_{t=1}^{\tau} H(\mathbf{P}_B^t).$$

Proof. Let \mathcal{C}_i be a client and for $j = 1, \dots, \beta$ and $t = 1, \dots, \tau$, let $X_{j, (h_j^t - 1)}^t$ be a set of visits from $h_j^t - 1$ clients other than \mathcal{C}_i to server \mathcal{S}_j in time frame t . We

have that

$$\begin{aligned}
H(\mathbf{C}_i) &\geq \sum_{t=1}^{\tau} H(\mathbf{C}_{i,B}^t | \mathbf{V}_B^{[t-1]}) \text{ (from Lemma 7)} \\
&\geq \sum_{t=1}^{\tau} H(\mathbf{C}_{i,B}^t | \mathbf{X}_{1,(h_1^t-1)}^t \cdots \mathbf{X}_{\beta,(h_\beta^t-1)}^t \mathbf{S}_B^t \mathbf{V}_B^{[t-1]}) \text{ (from (13) of Appendix A)} \\
&\geq \sum_{t=1}^{\tau} H(\mathbf{P}_B^t) \text{ (from Lemma 5).}
\end{aligned}$$

□

Notice that in Definition 1 we did not say anything on the entropies of random variables \mathbf{P}_j^t for $j \in \{1, \dots, m\}$ and $t \in \{1, \dots, \tau\}$. Our results apply to the general case of arbitrary entropies on proofs, but for clarity, we state the next corollary for the simpler case that $H(\mathbf{P}_{j_1}^{t_1}) = H(\mathbf{P}_{j_2}^{t_2})$ for all $j_1, j_2 \in \{1, \dots, m\}$ and $t_1, t_2 \in \{1, \dots, \tau\}$. We denote this common entropies by $H(\mathbf{P})$.

If the proof sequences of the corrupt servers are statistically independent, then the next corollary holds.

Corollary 9. *Let \mathcal{M} be an $(n, m, \tau, c, s, \{h_j\}_{j=1, \dots, m}^{t=1, \dots, \tau})$ dynamic multi-threshold metering scheme and let $\mathcal{S}_1, \dots, \mathcal{S}_s$ be the s corrupt servers. If the proof sequences of the s corrupt servers are statistically independent, then it holds that*

$$H(\mathbf{C}_i) \geq s\tau H(\mathbf{P}),$$

for any $i = 1, \dots, n$.

If the proofs for the servers are uniformly chosen in a finite field F , that is $H(\mathbf{P}) = \log |F|$, then from Corollary 9 and from (6) of Appendix A it holds that

$$\log |C_i| \geq s\tau \log |F|, \quad (5)$$

for any $i = 1, \dots, n$. This bound is tight, as in Section 3 we have presented a protocol for an $(n, m, \tau, c, s, \{h_j\}_{j=1, \dots, m}^{t=1, \dots, \tau})$ dynamic multi-threshold metering schemes which distributes *exactly* this information to clients.

5 Efficiency of the Scheme

In this section we analyze the efficiency of the scheme presented in Section 3. It is easy to see that the scheme meets the bounds (4) and (5) of Section 4. Indeed, during the initialization phase each client \mathcal{C}_i receives by the audit agency the polynomial $Q(i, y)$, which is of degree $s\tau - 1$. Therefore, the size of the information distributed to any client is $s\tau \log q$ and the bound (4) is tight. During a regular operation in a time frame t each client \mathcal{C}_i gives the value $Q(i, j \circ t)$ to the visited server \mathcal{S}_j . Therefore, the size of the information distributed to any visited server is $\log q$ and the bound (5) is tight. Hence, our protocol is optimal both with respect to the size of the information distributed to clients and with respect to the size of information given to servers by clients. This is important otherwise the task of receiving and sending information would burden the clients, that are not interested in the metering process.

6 Conclusions and Open Problems

In this paper we have introduced dynamic multi-threshold metering schemes. In these schemes the servers need to communicate with the audit agency at the beginning of any time frame.

In this paper we have assumed that clients provide correct values when they visit servers. In a practical implementation of a metering scheme, some method of authentication should be used. However, the method of authentication used would be, in general, not dependent on the specific metering scheme and it could be incorporated as an additional feature, if desired.

We have proved lower bounds on the size of the information distributed to clients and on the size of the information given from clients to servers during their visits. An interesting problem would be to provide lower bounds on the size of the information distributed to servers at the beginning of any time frame and to devise dynamic multi-threshold metering schemes in which this information is as small as possible.

Acknowledgements

This work was done while the third author was visiting the Department of Combinatorics and Optimization at the University of Waterloo, Ontario, Canada. She wants to thank the Department for its hospitality. The research of the fourth author is supported by the Natural Sciences and Research Council of Canada under grants NSERC-IRC 216431-96 and NSERC-RGPIN 203114-98.

A Information Theory Background

In this Appendix we review the basic concepts of Information Theory used in our definitions and proofs. For a complete treatment of the subject the reader is advised to consult [2].

Given a probability distribution $\{Pr_{\mathbf{X}}(x)\}_{x \in X}$ on a set X , we define the *entropy*¹ of \mathbf{X} , $H(\mathbf{X})$, as

$$H(\mathbf{X}) = - \sum_{x \in X} Pr_{\mathbf{X}}(x) \log Pr_{\mathbf{X}}(x).$$

The entropy satisfies the following property

$$0 \leq H(\mathbf{X}) \leq \log |X|, \tag{6}$$

where $H(\mathbf{X}) = 0$ if and only if there exists $x_0 \in X$ such that $Pr_{\mathbf{X}}(x_0) = 1$; whereas, $H(\mathbf{X}) = \log |X|$ if and only if $Pr_{\mathbf{X}}(x) = 1/|X|$ for all $x \in X$.

¹ All logarithms in this paper are to the base 2.

Given two sets X and Y and a joint probability distribution on their cartesian product, the *conditional entropy* $H(\mathbf{X}|\mathbf{Y})$, is defined as

$$H(\mathbf{X}|\mathbf{Y}) = - \sum_{y \in Y} \sum_{x \in X} Pr_{\mathbf{Y}}(y) Pr(x|y) \log Pr(x|y).$$

From the definition of conditional entropy it is easy to see that

$$H(\mathbf{X}|\mathbf{Y}) \geq 0. \quad (7)$$

The *mutual information* $I(\mathbf{X}; \mathbf{Y})$ between \mathbf{X} and \mathbf{Y} is defined by

$$I(\mathbf{X}; \mathbf{Y}) = H(\mathbf{X}) - H(\mathbf{X}|\mathbf{Y}) \quad (8)$$

and enjoys the following properties:

$$I(\mathbf{X}; \mathbf{Y}) = I(\mathbf{Y}; \mathbf{X}) \quad (9)$$

and $I(\mathbf{X}; \mathbf{Y}) \geq 0$, from which one gets

$$H(\mathbf{X}) \geq H(\mathbf{X}|\mathbf{Y}). \quad (10)$$

Given three sets X, Y, Z and a joint probability distribution on their cartesian product, the *conditional mutual information* $I(\mathbf{X}; \mathbf{Y}|\mathbf{Z})$ between \mathbf{X} and \mathbf{Y} given \mathbf{Z} is

$$I(\mathbf{X}; \mathbf{Y}|\mathbf{Z}) = H(\mathbf{X}|\mathbf{Z}) - H(\mathbf{X}|\mathbf{Z}\mathbf{Y}) \quad (11)$$

and enjoys the following properties:

$$I(\mathbf{X}; \mathbf{Y}|\mathbf{Z}) = I(\mathbf{Y}; \mathbf{X}|\mathbf{Z}) \quad (12)$$

and $I(\mathbf{X}; \mathbf{Y}|\mathbf{Z}) \geq 0$. Since the conditional mutual information is always non negative we get

$$H(\mathbf{X}|\mathbf{Z}) \geq H(\mathbf{X}|\mathbf{Z}\mathbf{Y}). \quad (13)$$

Given $n + 1$ sets X_1, \dots, X_n, Y and a joint probability distribution on their cartesian product, the entropy of $\mathbf{X}_1 \dots \mathbf{X}_n$ given \mathbf{Y} can be expressed as

$$H(\mathbf{X}_1 \dots \mathbf{X}_n|\mathbf{Y}) = H(\mathbf{X}_1|\mathbf{Y}) + \sum_{i=2}^n H(\mathbf{X}_i|\mathbf{X}_1 \dots \mathbf{X}_{i-1} \mathbf{Y}) \quad (14)$$

and enjoys the following property:

$$H(\mathbf{X}_1 \mathbf{X}_2 \dots \mathbf{X}_n|\mathbf{Y}) \leq \sum_{i=1}^n H(\mathbf{X}_i|\mathbf{Y}). \quad (15)$$

B Parameters and Variables Used in the Paper

| | |
|---------------------------|---|
| n | number of clients |
| m | number of servers |
| τ | number of time frames |
| c | number of corrupt clients |
| s | number of corrupt servers |
| h_j^t | threshold for server \mathcal{S}_j in time frame t |
| \mathbf{C}_i | information distributed to client \mathcal{C}_i |
| $\mathbf{C}_{i,j}^t$ | visit from client \mathcal{C}_i to server \mathcal{S}_j in time frame t |
| $B = \{1, \dots, \beta\}$ | indices of corrupt servers, $\beta \leq s$ |
| $\mathbf{C}_{i,B}^t$ | visits from client \mathcal{C}_i to servers $\mathcal{S}_1, \dots, \mathcal{S}_\beta$ in time frame t |
| $\mathbf{X}_{j,(d_j)}^t$ | visits from d_j clients to server \mathcal{S}_j in time frame t |
| \mathbf{S}_j^t | information distributed to server \mathcal{S}_j at the beginning of time frame t |
| \mathbf{S}_B^t | information distributed to servers $\mathcal{S}_1, \dots, \mathcal{S}_\beta$ at the beginning of time frame t |
| \mathbf{P}_j^t | proof for server \mathcal{S}_j in time frame t |
| \mathbf{P}_B^t | proofs for servers $\mathcal{S}_1, \dots, \mathcal{S}_\beta$ in time frame t |
| $\mathbf{V}_j^{[t]}$ | information collected by server \mathcal{S}_j in time frames $1, \dots, t$ |
| $\mathbf{V}_B^{[t]}$ | information collected by servers $\mathcal{S}_1, \dots, \mathcal{S}_\beta$ in time frames $1, \dots, t$ |

References

1. C. Blundo, A. De Bonis and B. Masucci, *Metering Schemes with Pricing*, to appear in Proceedings of “14th International Symposium on DIStributed Computing - DISC 2000”, Toledo, Spain, October 4–6, 2000.
2. T. M. Cover and J. A. Thomas, *Elements of Information Theory*. John Wiley & Sons, 1991.
3. A. De Bonis and B. Masucci, *An Information Theoretical Approach to Metering Schemes*, in Proceedings of “2000 International Symposium on Information Theory - ISIT 2000”, Sorrento, Italy, June 25–30, 2000.
4. M. Franklin and D. Malkhi, *Auditable Metering with Lightweight Security*, in Proceedings of “Financial Cryptography ’97”, Lecture Notes in Computer Science, Vol. **1318**, pp. 151–160, 1997.
5. B. Masucci and D. R. Stinson *Efficient Metering Schemes with Pricing*, submitted for publication, 2000. Technical Report CORR **2000-06**, Centre for Applied Cryptographic Research, University of Waterloo.
6. B. Masucci and D. R. Stinson, *Metering Schemes for General Access Structures*, to appear in Proceedings of “6th European Symposium on Research in Computer Security - ESORICS 2000”, Toulouse, France, October 4–6, 2000. Technical Report, CORR **2000-21**, Centre for Applied Cryptographic Research, University of Waterloo.

7. M. Naor and B. Pinkas, *Secure and Efficient Metering*, in Proceedings of “Advances in Cryptology - EUROCRYPT '98”, Lecture Notes in Computer Science, Vol. **1403**, pp. 576–590, 1998.
8. M. Naor and B. Pinkas, *Secure Accounting and Auditing on the Web*, Computer Networks and ISDN Systems, Vol. **40**, Issues 1-7, pp. 541–550, 1998.
9. A. Shamir, *How to Share a Secret*, Communications of the ACM, Vol. **22**, pp. 612–613, 1979.

Chained Stream Authentication^{*}

Francesco Bergadano¹, Davide Cavagnino¹, and Bruno Crispo^{2,**}

¹ Computer Science Department, University of Turin,
Corso Svizzera 185, 10149 Torino, Italy
{`bergadan,davide`}@di.unito.it

² SRI International, Information Assurance Lab Cambridge,
23 Millers Yard, Mill Lane, Cambridge CB2 1RQ, UK
`crispo@cam.sri.com`

Abstract. We present a protocol for the exchange of individually authenticated data streams among N parties. Our authentication procedure is fast, because it only requires the computation of hash functions – we do not need digital signatures, that are substantially less efficient. The authentication information is also short: two hash values for every block of data. Since there are no shared secrets, this information does not grow with N , the number of parties.

1 Introduction

Multicast applications are receiving increasing commercial interest. In particular, multicast conferencing tools are available that support real-time multiway communications over a packet network.

The security of multicast and videoconferencing has also become an important and specific issue [8,3,21], and it includes two forms of authentication services:

- *Group authentication.* The conference participants share one key, and authentication allows to check that data has not been modified or inserted by attackers *outside* the group [21,24,3].
- *Individual authentication.* Received data streams must be authenticated with respect to their individual origin, i.e. one individual group member [8].

This paper addresses individual authentication. An authentication protocol will be obtained, that is secure even when attackers may adaptively choose authenticated data streams. For an interactive scenario, that is typical of multicast conferencing, the proposed solution is substantially more efficient than in previous approaches.

^{*} The protocol described in this paper, and the timing protocol derived from it, were first presented in a seminar at IBM T. J. Watson Research Center in summer 1998 by F. Bergadano

^{**} This work was completed while Bruno Crispo was with the Computer Science Department, University of Turin

2 Previous Work

2.1 Individual Authentication in Multicast Groups

Individual authentication may be essential in multicast applications. Yet, it is difficult to achieve. If one uses digital signatures, the required computational cost may be too high. In fact, end-user hardware may be limited and loaded by the multimedia processing that is part of the conferencing application. If one uses symmetric Message Authentication Codes (MACs), every stream block will need to carry a distinguished code for every recipient. Previous work on individual authentication for multicast is all based on one of the above schemes (signature or multiple MACs), with modifications directed to improve efficiency.

On the side of symmetric authentication, it is possible to use a fixed number K of MAC keys, that does not depend on the size N of the multicast group [8,10]. The sender knows K keys, and each receiver knows $K/2$ keys, chosen at random. Each stream block is authenticated with K MACs, corresponding to the K keys, and receivers check the MACs for which they have a key. Obviously, receiver collusions can lead to forged individual authentication codes.

More work is available on the side of efficient digital signatures. Online/offline signatures [11] may be used to split the computation in a more expensive off-line phase, followed by an efficient online phase that is needed when the data becomes available. On the other hand, part of the signature can be performed by a “signature server”, that may be located elsewhere, without compromising authentication or even non-repudiation [2]. One-time signatures [18,7] are an efficient alternative, and are adequate for stream authentication. Gennaro and Rohatgi [12] have proposed an efficient stream signing method that is based on a chain of one-time signatures. The disadvantage of this approach lies in the length of one-time signatures.

2.2 Hash Chains

Our work does not fall within the two above categories. In particular, it is not based on asymmetric cryptography, and there are no shared secrets. We use single individual MACs, and a hash chain of individual, secret keys. Hash chains have been used for a long time and for a number of different purposes.

Hash chains were first proposed by Lamport [17] and then used in the S/Key [15] user identification system. They have been applied to the authentication of public key certificate revocation/validity messages [20], to digital payment systems [23], and to Web server hit acknowledgements. All of the above applications basically use hash chains to send periodic “yes, I’m alive” messages in a secure way. In other words, they are *contentless*, meaning that content is bound to the chain at the start, but is not added or modified when the individual hash values are released.

This is not the case with more recent approaches, where hash chains have been proposed for the authentication of messages, as described in [2,1]. The basic idea of [1] is to MAC each block of data with a new key, and send the key as part

of the next message. The main problem with that technique is that when a key is sent too early (i.e. before the previous message was obtained by all intended recipients), falsifying the rest of the stream becomes possible.

A time protocol based on hash chains was developed independently by [9] and by [5]. This protocol was also published in [22] and [6].

In this paper we present an interactive protocol that does not depend on time, and provide proofs of its security.

3 Chained Stream Authentication

Following the formalization of [13], and [12], we define a *security parameter*, n , and say that a function $\epsilon(n)$ is “negligible”, if, for all constants c , there is n_0 such that, for $n > n_0$, $\epsilon(n) < 1/n^c$.

Following [13], we define a signature scheme as a triple (G, Sig, V) of probabilistic polynomial time algorithms, where (1) G is used to generate a key pair (SK, PK) , (2) Sig is used to sign any message M , using the secret key SK , and (3) V is the signature verification algorithm, such that $V(PK, M, \text{Sig}(SK, M)) = 1$, for any message M . We will use a signature scheme that is secure against adaptively chosen message attacks [13]: the probability of forging a signature is negligible, even when a signature oracle is available.

Similarly, we define a *stream authentication scheme* as a triple of probabilistic polynomial-time algorithms (GA, AA, VA) , where

- On input 1^n , GA outputs a pair of keys $(SK, PK) \in \{0, 1\}^{2n}$.
- AA is the authentication algorithm, and receives in input a secret key SK , and a stream $S = S_1, S_2, \dots, S_i$, consisting of a finite number i of *blocks*. AA outputs an authenticated stream $S' = S'_1, \dots, S'_i$, where $S'_j = (S_j, \text{auth}_j)$, being auth_j some kind of authentication data.
- The verification algorithm VA is such that $VA(PK, AA(SK, S)) = 1$. When $VA(PK, S') = 1$, we will say that S' is *valid*.

We may now define our proposed scheme, called a *chained stream authentication scheme*:

- As a generator GA , we use the generator of a signature scheme (G, Sig, V) , secure against chosen message attacks.
- The authentication algorithm AA will be called a “Chained Stream Authentication” algorithm (CSA). This algorithm first generates a secret α , computes $h^k(\alpha)$ for some $k > i$, and then produces the following output:

$$\begin{aligned} S'_1 &= S_1, \text{MAC}_{h^{k-1}(\alpha)}(S_1), h^k(\alpha), SN, \text{Sig}(SK, h^k(\alpha), SN) \\ S'_2 &= S_2, \text{MAC}_{h^{k-2}(\alpha)}(S_2), h^{k-1}(\alpha) \\ &\dots\dots\dots \\ S'_i &= S_i, \text{MAC}_{h^{k-i}(\alpha)}(S_i), h^{k-i+1}(\alpha) \end{aligned}$$

By MAC, we denote a secure Message Authentication Code, i.e. such that the probability of forging a valid code is negligible, even when a MAC oracle is available. For h , we will use a collision resistant hash function. The

- authentication includes a session number SN that is incremented for every new stream.
- The verification algorithm VA will output 1 if the initial asymmetric signature is valid, if all the MACs are correct, and if the hash chain of the MAC keys is consistent, i.e. the hash of a key produces the previous key in the chain.

4 Security against Continuations

What are the security properties of the proposed scheme? Clearly, given a stream authentication oracle, it is possible to forge new valid authenticated streams, because the MAC keys become known. Therefore, CSA is not “secure” according to the definition of [12], that can be rephrased as follows: “a stream authentication scheme (GA, AA, VA) is secure if any probabilistic polynomial-time algorithm F , given as input the public key PK and adaptively chosen authenticated streams $S^{(j)}$, outputs a new valid authenticated stream $S' \not\subseteq S^{(j)}$, for all j , only with negligible probability”. Clearly, this definition of security does not apply to the CSA scheme: the forger F may ask for just one authenticated stream, change any block but the last, and recompute the corresponding MAC using the available key.

However, our scheme satisfies a weaker security notion, which we will call “security against continuations”. We define continuations as follows:

Definition. A stream S_2 is a **continuation** of a stream S_1 , denoted by $S_1 \subset S_2$, if S_1 is a proper prefix of S_2 .

The same definition applies to authenticated streams under (GA, AA, VA) . A valid authenticated continuation S'_2 of an authenticated stream S'_1 must then be such that $S'_1 \subset S'_2$ and $VA(PK, S'_2) = 1$. Security against continuations means, informally, that it is unfeasible to produce valid continuations of observed valid streams. This weaker notion will nevertheless be sufficient for building secure authentication protocols, after some means of sender/receiver synchronization is achieved, as described in Sections 5 and 6. More precisely, security against continuations corresponds to the following:

Intuition. A forger may produce a new valid stream S only if it is associated to a stream T , that was used previously. Moreover, if $|S| = |T|$, and the last blocks of S and T are different, then it will be impossible to produce a valid continuation of S .

Next, we formalize the above intuition and prove that it applies to CSA (in Lemma 1, with proof given in Appendix A).

Definition. A stream authentication scheme is **secure against continuations** if there is no polynomial time algorithm F that, given adaptively chosen authenticated streams $S^{(1)}, \dots, S^{(k)}$, is able to generate a valid authenticated stream $S' = S'_1, \dots, S'_j$ with non-negligible probability, unless $\exists i \in [1, k]$ such that $S_1^{(i)} = S_1^{(i)}$, MAC_1^i , H , SN , $sig(H, SN)$, where $S'_1 = S_1$, MAC_1 , H , SN , $sig(H, SN)$, and one of the following holds:

1. $j < |S^{(i)}|$, or

2. $j = |S'^{(i)}|$, and $S'_j = S_j'^{(i)}$, or
3. $j = |S'^{(i)}|$, and $S'_j \neq S_j'^{(i)}$, and there is no polynomial time algorithm F' that can generate with non-negligible probability a valid continuation $T' \supset S'$, given $S'^{(1)}, \dots, S'^{(k)}$, possibly other adaptively chosen authenticated streams, and any valid authenticated continuation of $S'^{(i)}$.

Security against continuations is a complicated notion, but it will lead us to a simple concept of stream authentication in Theorem 1. First, though, we need the following:

Lemma 1. *Suppose that, in the CSA authentication scheme,*

- (1) (G, S, V) is a secure signature scheme,
 - (2) g is a pseudorandom function,
 - (3) $h(x) = g_x(0)$ and $MAC_k(x) = g_k(1, x)$
 - (4) g is such that h is a collision resistant hash function.
- Then, the scheme (GA, CSA, VA) is secure against continuations.*

MAC and h are of the CSA algorithm, and are defined through a pseudorandom function [14,4] as defined in (3) above, because not only should the MAC be secure, but each key k must look random even though $h(k)$ is known. With the definition of (3), knowing $h(k) = g_k(0)$ gives no additional information, as one could in any case query the oracle for g_k and obtain $g_k(0)$. In practice, one could use $g = \text{HMAC}$ [16] so as to satisfy both (2) and (4).

5 The Chained Stream Authentication Protocol with One Sender and One Receiver

We will now use the CSA scheme to authenticate information over an insecure network. In fact, CSA's security against continuations can be used with a synchronization mechanism to obtain a very efficient individual authentication method. For now, we consider one party, named A, who will send authenticated data, and one party, named B, who will receive the data. The protocol is defined below, where $Sig_A(x) = Sig(SK_A, x)$ is A's signature under (G, S, V) , and similarly $Sig_B(x) = Sig(SK_B, x)$ for B:

1. B \rightarrow A: $h^k(\beta), SN, Sig_B(h^k(\beta), SN)$
 A \rightarrow B: $A_1, MAC_{h^{k-1}(\alpha)}(A_1), h^k(\alpha), SN, Sig_A(h^k(\alpha), SN)$
2. B \rightarrow A: $h^{k-1}(\beta)$
 A \rightarrow B: $A_2, MAC_{h^{k-2}(\alpha)}(A_2), h^{k-1}(\alpha)$
- ...
- i. B \rightarrow A: $h^{k-i+1}(\beta)$
 A \rightarrow B: $A_i, MAC_{h^{k-i}(\alpha)}(A_i), h^{k-i+1}(\alpha)$
- ...

Messages are sequential: A will not send message i if it has not received a correct i -th message from B, and B will not send message $i + 1$ if it has not received from A a correct i -th message. A and B initially generate individual

random secrets α and β , and compute $h^k(\alpha)$ and $h^k(\beta)$, respectively. These values, and the session number SN, are signed, and exchanged as part of the first messages in step 1. Then, A sends data as defined in the CSA scheme, and B sends back authenticated acknowledgments. B's authenticated ack for A's j -th message is simply $h^{k-j}(\beta)$. The receiver side is then similar to what happens in S/Key and similar applications [15,17]. The security properties of the protocol, to be discussed next, are made relative to the following:

Active Attack Model. CSA sender A, CSA receiver B, and attacker E:

- *E runs in polynomial time and may ask A to send B the authenticated streams $S^{(1)}, \dots, S^{(k)}$ in sessions $1, \dots, k$;*
- *A chooses stream $S^{(k+1)}$ and sends it to B in session $k + 1$;*
- *At any time, E can read messages, stop messages and insert messages;*
- *During session $k + 1$, E tries to have B receive $S' \neq S^{(k+1)}$ and believe it authentic.*

We call the above an *active stream authentication attack*. We shall prove in Theorem 1 that such attacks are not feasible with the above CSA protocol, except for the possible falsification of the last block of $S^{(k+1)}$. We first note that session numberings by sender and receiver are consistent:

Observation 1. *Suppose A has sent its first message of session SN. Then B must have already sent its first message of session SN.*

Observation 2. *Suppose B has sent its second message of session SN. Then A must have already sent its first message of session SN.*

The observations allow us to speak of a “current session” in the CSA protocol with one sender and one receiver. We can now prove that this protocol represents a valid authentication mechanism:

Theorem 1 (Security of CSA with one sender and one receiver). *Suppose sender A and receiver B run SN sessions under the CSA protocol, where:*

- *the conditions of Lemma 1 hold;*
- *a polynomial active attacker E chooses streams $A^{(1)}, \dots, A^{(SN-1)}$, that A sends to B in sessions $1, \dots, SN - 1$;*
- *B has received the valid authenticated stream $S' = S'_1, \dots, S'_j$ during session SN.*

Then, E can cause S'_1, \dots, S'_{j-1} to be non-authentic only with negligible probability.

The proof is by induction on $|S'|$, based on Lemma 1 (see Appendix A). The last block of S' may be modified by E. Thus, authentication is obtained with a one block delay: the receiver can ascertain the origin and the integrity of a block in the stream only after receiving the next block. This is acceptable in most multicast applications. This is achieved without shared secrets and without signatures after the first message. The important consequences of this fact are discussed in the next session, where the protocol is used with more than just two parties.

6 The N-Party CSA Protocol

We will now extend the protocol so that it can be used effectively in a multicast conferencing scenario. As a first step, we will consider two parties, A and B, that must exchange authenticated data in both directions. Obviously, this can be done by simply applying the CSA protocol with sender A and receiver B, and simultaneously with sender B and receiver A. However, we can make this more efficient by merging the hash values sent as acknowledgments and the ones sent as hashes of secrets. This results in the following **two-party protocol**:

1. B \rightarrow A: $B_1, MAC_{h^{k-1}(\beta)}(B_1), h^k(\beta), SN, Sig_B(h^k(\beta), SN)$
 A \rightarrow B: $A_1, MAC_{h^{k-1}(\alpha)}(A_1), h^k(\alpha), SN, Sig_A(h^k(\alpha), SN)$
2. B \rightarrow A: $B_2, MAC_{h^{k-2}(\beta)}(B_2), h^{k-1}(\beta)$
 A \rightarrow B: $A_2, MAC_{h^{k-2}(\alpha)}(A_2), h^{k-1}(\alpha)$
- ...
- i. B \rightarrow A: $B_i, MAC_{h^{k-i}(\beta)}(B_i), h^{k-i+1}(\beta)$
 A \rightarrow B: $A_i, MAC_{h^{k-i}(\alpha)}(A_i), h^{k-i+1}(\alpha)$
- ...

The above protocol may cause practical transmission difficulties. In particular, each party may send a block of data only after receiving a corresponding block from the other party. This causes a kind of stop-and-wait behaviour that implies poor network utilization and may result in unacceptable delays for real-time traffic. Fortunately, such strict sequentialization of messages is not necessary. In particular, data blocks and MACs can be sent at any time - only the delivery of keys need be delayed until acknowledgements are obtained and verified. We may therefore rewrite the above two party protocol by splitting the behaviour of each party into a *data sender* process and a *key sender* process, that run independently. For party A, the processes are defined as follows (party B is defined symmetrically):

A's data sender process:

1. send to B: $MAC_{h^{k-1}(\alpha)}(A_1), A_1$
2. send to B: $MAC_{h^{k-2}(\alpha)}(A_2), A_2$
- ...

A's key sender process:

1. wait for $MAC_{h^{k-1}(\beta)}(B_1)$
 send to B: $h^k(\alpha), SN, sig_A(h^k(\alpha), SN)$
2. wait for $MAC_{h^{k-2}(\beta)}(B_2)$
 wait for $h^k(\beta), SN, sig_B(h^k(\beta), SN)$
 send to B: $h^{k-1}(\alpha)$
3. wait for $MAC_{h^{k-3}(\beta)}(B_3)$
 wait for $h^{k-1}(\beta)$
 send to B: $h^{k-2}(\alpha)$
- ...

Delaying just secrets, not information, is essential in multicast conferencing: the receiver will continue viewing the stream of data even though the keys necessary to authenticate it are not yet available. When secrets are late, viewing is ahead of authentication, and we call this an *authentication delay*. The delay would be small and roughly equivalent to three times the network latency. The reason is that a MAC must be sent, then the authenticated acknowledgement is returned, and finally the MAC key is sent. Only then can the corresponding block be authenticated by the receiver.

We may now generalize the above construction and obtain the **N-party protocol**. During session SN , party i first generates a random secret α_i and computes $h^k(\alpha_i)$. Party i consists of two processes, a *data sender* and a *key sender*, that run concurrently as described below, where $A_{i,j}$ is the j th block sent by party i :

Data sender i :

1. multicast $MAC_{h^{k-1}(\alpha_i)}(A_{i,1})$ and $A_{i,1}$;
2. multicast $MAC_{h^{k-2}(\alpha_i)}(A_{i,2})$ and $A_{i,2}$;
- ...

Key sender i :

1. wait for $MAC_{h^{k-1}(\alpha_j)}(A_{j,1})$, for all $j \in [1, N]$;
multicast $h^k(\alpha_i)$, $SN, i, sig_i(h^k(\alpha_i), SN, i)$;
2. wait for $MAC_{h^{k-2}(\alpha_j)}(A_{j,2})$, for all $j \in [1, N]$;
wait for $h^k(\alpha_j)$, $SN, j, sig_j(h^k(\alpha_j), SN, j)$, for all $j \in [1, N]$;
multicast $h^{k-1}(\alpha_i)$;
3. wait for $MAC_{h^{k-3}(\alpha_j)}(A_{j,3})$, for all $j \in [1, N]$;
wait for $h^{k-1}(\alpha_j)$, for all $j \in [1, N]$;
multicast $h^{k-2}(\alpha_i)$;
- ...

It is important to note that, for every block, the only authentication information that is multicast is one MAC (sent by the data sender) and one hash value (sent by the key sender). This does not grow with N .

Conclusions

We have defined a Chained Stream Authentication scheme and proved it to be secure against non-authentic stream continuations. This property was the basis for an interactive stream authentication protocol that was proven to be secure against active attacks, even when the attacker may ask for the authentication of a number of adaptively chosen streams. The protocol was then optimized for the case of a bidirectional flow of data.

However, the importance of the protocol arises when there are more than just two communicating parties. Since there are no shared secrets, the size of the authentication data does not grow linearly with the number of parties. In

the optimized protocol for N parties, we need just one MAC and one hash value for every block of data. By contrast, symmetric individual authentication would require N distinguished MACs for every block. Therefore, the CSA protocol represents a major improvement. If compared with techniques based on the digital signature of each block, CSA is to be preferred because of its much higher efficiency.

With respect to the work in [12], our scheme has the advantage that if a packet is lost, using a timeout when waiting for all the MACs and keys, the authentication may proceed for the following blocks due to the use of a chain of keys, while in [12] a packet may be verified only if all the preceding packets have been received. Moreover, the online solution in [12] uses one-time signatures, that introduce a communication overhead of the order of 1000 bytes per packet. The CSA solution is an order of magnitude more efficient than [12] with respect to the authentication information transmitted by the sender. Another difference with our work is that [12] allows non-repudiation, while our scheme does not.

Appendix A (Proof of Lemma 1 and Theorem 1)

Proof of Lemma 1. Suppose that a forger F exists that can produce a valid authenticated stream S' with a non-negligible probability ϵ , contraddicting the thesis. Then, one of the two following cases must hold, and at least one must hold with probability at least $\epsilon/2$:

Case 1 $S'_1 = S_1, MAC_1, H, SN, sig(H, SN)$ and there is no $S'^{(i)}$ such that $S'^{(i)}_1 = S^{(i)}_1, MAC^{(i)}_1, H, SN, sig(H, SN)$. Then, we can use the forger F to construct algorithm $F1$ that breaks the asymmetric signature scheme (G, S, V) . The constructed algorithm $F1$ has access to an oracle for S , and starts by calling F as a subroutine. When F requires an authenticated stream $S'^{(i)}$, $F1$ generates a random secret $\alpha^{(i)}$, computes $h^k(\alpha^{(i)})$, and asks the oracle for $sig(h^k(\alpha^{(i)}), SN)$. Then, knowing $\alpha^{(i)}$, $F1$ authenticates the rest of the stream as required by CSA, and outputs the authenticated stream $S'^{(i)}$ for F . The process continues until, with probability at least $\epsilon/2$, F outputs the new valid authenticated stream S' . In this case (Case 1), S' must be such that $S'_1 = S_1, MAC_1, H, SN, sig(H, SN)$ and there is no $S'^{(i)}$ generated by $F1$ such that $S'^{(i)}_1 = S^{(i)}_1, MAC^{(i)}_1, H, SN, sig(H, SN)$. This means that a signature $sig(H, SN)$ was never queried by $F1$ to the oracle. Hence $F1$ breaks (S, G, V) by outputting the new valid signature $H, SN, sig(H, SN)$.

Case 2 $S'_1 = S_1, MAC_1, H, SN, sig(H, SN)$ and there is $S'^{(i)}$ such that $S'^{(i)}_1 = S^{(i)}_1, MAC^{(i)}_1, H, SN, sig(H, SN)$. Then there are three cases, and one must hold with probability at least $\epsilon/6$:

Case 2.1 $|S'| < |S'^{(i)}|$. This case does not contraddict the Lemma.

Case 2.2 $|S'| > |S'^{(i)}|$. In this case we can construct $F2$ that can invert h , using the forger F , and hence break g . The inverter $F2$ is given a value α and must compute $h^{-1}(\alpha)$ with non-negligible probability. $F2$ calls GA to generate a key pair (SK, PK) , and then runs F as a subroutine. Let SN_{max} be the maximum

number of authenticated streams that F will ask for. Clearly SN_{max} must be polynomially large. F2 will then pick a random number R between 1 and SN_{max} . When asked to authenticate stream $S^{(R)}$, F2 lets $\alpha^{(R)} = \alpha$ and then follows the CSA authentication algorithm, but choses $k = |S^{(R)}|$. When F stops, it will output S' such that $|S'| > |S'^{(i)}|$, with probability greater than $\epsilon/6$, and $i = R$ with probability $1/SN_{max}$. Hence, with probability greater than $\epsilon/(6 * SN_{max})$, $S'_{|S'^{(i)}|+1} = (S, MAC, h^{k-(|S'^{(i)}|+1)}(\alpha)) = (S, MAC, h^{k-(k+1)}(\alpha)) = (S, MAC, h^{-1}(\alpha))$. Since $h(x) = g_x(0)$, inverting h means breaking g_{key} after observing $g_{key}(0)$.

Case 2.3 $|S'| = |S'^{(i)}| = j$. There are two cases, and one must hold with probability at least $\epsilon/12$:

Case 2.3.1 $S_j = S_j^{(i)}$. This case does not contradict the Lemma.

Case 2.3.2 $S_j \neq S_j^{(i)}$. Let $S'_j = S_j, MAC_{key}(S_j), h^{k-j+1}(\alpha)$. There are two cases, and at least one must occur with probability at least $\epsilon/24$:

Case 2.3.2.1 $MAC_{key}(S_j)$, generated by F, and $MAC(S_j^{(i)})$, given in stream $S'^{(i)}$, are valid under the same key. In this case we can use F to break g_{uk} , for some unknown key uk , in a polynomial algorithm F3, that has access to an oracle for g_{uk} . Define again SN_{max} as the maximum number of authenticated streams that F will ask for. F3 will then pick a random number R between 1 and SN_{max} . F3 will run F as a subroutine, will use GA to generate a key pair (SK, PK) , and will authenticate all streams requested by F normally using CSA, except for stream $S^{(R)}$. For this stream, define $l = |S^{(R)}|$, and let $\alpha^{(R)} = uk$, and $k = l$. Then, $h^{k-l+1}(\alpha^{(R)}) = h(uk)$. F3 then computes $h^{l-1}(uk), \dots, h(uk)$, after querying the oracle for $h(uk) = g_{uk}(0)$, and uses these values, in this order, to compute the MACs for $S_1^{(R)}, \dots, S_{l-1}^{(R)}$, as required in CSA. As the last authenticated block, F3 outputs $S'_l = (S_l^{(R)}, MAC_{uk}(S_l^{(R)}), h(uk))$, where $MAC_{uk}(S_l^{(R)}) = g_{uk}(1, S_l^{(R)})$, is queried to the oracle. With probability $1/SN_{max}$, the authenticated stream S' output by F is associated to stream $S^{(R)}$, i.e., $i = R$. In this case, $MAC_{uk}(S_j^{(R)})$ and $MAC_{key}(S_j)$, are valid under the same key, i.e., $key = uk$. F3 then outputs $MAC_{uk}(S_j) = g_{uk}(1, S_j)$, and since $S_j \neq S_j^{(i)}$, this is a new forged MAC, and a correct value of g_{uk} for the new input $(1, S_j)$, that is generated with non-negligible probability greater than $\epsilon/(24 * SN_{max})$.

Case 2.3.2.2 $MAC_{key}(S_j)$, generated by F, and $MAC(S_j^{(i)})$, given in stream $S'^{(i)}$, are not valid under the same key. We show that, in this case, there is no polynomial time algorithm F' that can generate with non-negligible probability a valid authenticated continuation $T' \supset S'$, given $S'^{(1)}, \dots, S'^{(k)}$, possibly other adaptively chosen authenticated streams $T'^{(1)}, \dots, T'^{(p)}$, and any valid continuation of $S'^{(i)}$. Suppose such a forger F' exists, and does the above with non-negligible probability ϵ' . Let $T'_{j+1} = (T_{j+1}, MAC, key)$. Since T' is valid and is a continuation of S' , we also know that $T'_j = S'_j = (S_j, MAC_{key}(S_j), h^{k-j+1}(\alpha))$. We construct F4 that can generate collisions for h with non-negligible probability, using F and F'. F4 starts by generating a key pair (SK, PK) . Then, it runs

F as a subroutine and authenticates the requested streams normally using CSA. F will then output S' satisfying the conditions of this case. We also know that a continuation forger F' exists and therefore S' has a possible valid continuation. Consequently $MAC_{key}(S_j)$ is a valid MAC for some value of key , and for the conditions in this case, $key \neq h^{k-j}(\alpha)$. This all happens with probability greater than $\epsilon/24$. In order to obtain key , and thus obtain a collision for h , F4 must then run F' as a subroutine. F4 will authenticate additional streams asked by F' normally, using CSA, and will then produce a continuation of $S'^{(i)}$, as requested by F' , also using CSA. When F' outputs a valid continuation T' of S' , this must include the value of key , and F4 outputs $(key, h^{k-j}(\alpha))$ as a collision for h . This must occur with non-negligible probability (greater than $\epsilon * \epsilon'/24$). QED

Proof of Observation 1: We construct an algorithm F that simulates A and B over an insecure network, where E can perform active attacks. F controls the simulations of A and B out of band, i.e. over a secure, separate channel. Suppose that, with non-negligible probability, E has taken action so that B has not yet sent the first message of session SN , when A has already sent its first message of session SN . Then we construct F so that it can forge signatures under the asymmetric scheme (G, S, V) . F simulates A normally, by first calling GA to generate a key pair (SK_A, PK_A) . For simulating B, F does not generate a key pair, but relies on the oracle for the signature required in the first message of every session. At some point, F's simulation of A must have computed the first message of session SN , and it must therefore have received $sig_B(H, SN)$. However, we have supposed F's simulation of B has not yet computed the first message of session SN . As a consequence, $sig_B(H, SN)$ was never queried to the oracle, and can be output as a forged signature. QED.

Proof of Observation 2: Under the same setting of Observation 1, F simulates B normally, by first calling GA to generate a key pair (SK_B, PK_B) . For simulating A, F does not generate a key pair, but relies on the oracle for the signature required in the first message of every session. At some point F's simulation of B must have sent the second message of session SN , and therefore it must have received $sig_A(H, SN)$. However, since F's simulation of A has not yet begun running session SN , it has not yet computed the first authenticated block. As a consequence, $sig_A(H, SN)$ was never queried to the oracle, and can be output as a forged signature. QED.

We shall now prove Theorem 1 by induction on $|S'|$, based on Lemma 1. However, we first need the following, that characterizes the information available to an active attacker at any given moment:

Lemma 2. *Suppose A and B run session SN , where:*

- (1) (G, S, V) is a secure signature scheme, and
- (2) h is a one-way hash function

Suppose also that B has received, during session SN , the valid authenticated stream $S' = S'_1, \dots, S'_{j-1}$, and no more. Then, there is no active attacker E that can, with non-negligible probability, cause A to release more than j authenticated blocks during session SN .

Proof. Let parties A and B run session SN of the CSA protocol with one sender and one receiver. Suppose the Lemma is false. Then there is an active attacker E that can, with non-negligible probability, cause a situation where A has released the stream $A' = A'_1, \dots, A'_{j+1}$, while B has only received $j - 1$ blocks S'_1, \dots, S'_{j-1} . Since A has released $j + 1$ blocks, it must have received authenticated acknowledgements $h^k(\beta), \dots, h^{k-j}(\beta)$. There are two cases, and one must hold with probability at least $\epsilon/2$:

Case 1: $\beta \neq \beta^{(SN)}$, the secret value chosen by B for session SN. Then, we show that we can construct an algorithm F1 that can simulate A and B, and use a signature oracle to forge signatures under (G,S,V). F1 simulates A by running the sender side of the CSA protocol. F1 simulates B by running the receiver side of the CSA, but uses the signature oracle to produce the signatures needed in the first authenticated block of each session. When E has caused the situation covered by this case, F1's simulation of A must have received $sig(\beta, SN)$, and since $\beta \neq \beta^{(SN)}$, this can be output as a new forged signature.

Case 2: $\beta = \beta^{(SN)}$. Then, we can construct F2 that can compute $h^{-1}(x)$, for any x , with non-negligible probability. F2 will simulate A and B, running the CSA protocol over a network where E can perform active attacks. F2 simulates A by running the sender side of the CSA protocol. F2 simulates B by running the receiver side of the CSA, but first sets the following values:

- pick SN at random between 1 and SN_{max} , where SN_{max} is the maximum number of sessions that the attacker is able to cover;
- pick j at random between 1 and j_{max} , where j_{max} is the maximum number of blocks per session in E's active attacks;
- let $k = j - 1$ and $\beta^{(SN)} = h^j(x)$;

B is then able to send to A all acknowledgments required for receiving the first $j - 1$ blocks, i.e., $h^k(\beta^{(SN)}) = h^{2j-1}(x), \dots, h^{k-(j-1)}(\beta^{(SN)}) = x$. When E, has caused the situation covered by this case, F2's simulation of A must have received $h^{k-j}(\beta^{(SN)}) = h^{-1}(x)$. This happens with non-negligible probability $\epsilon/2j_{max}SN_{max}$. QED.

Proof of Theorem 1: We prove a stronger claim by induction on $|S'| = j$, namely: under the conditions of the Theorem, the thesis holds and, if S'_j is non-authentic, then E can cause B to receive a valid continuation of S' only with negligible probability.

Base: We show that the claim is true for $j = 1$. Before B has received from A the first message of session SN, by Lemma 2, A has released no more than the first block of session SN. Therefore, the information available to E consists of:

- the previous authenticated streams $A^{(1)}, \dots, A^{(SN-1)}$ sent by A, and
- the first block $A_1^{(SN)}$ of session SN.

Let $S'_1 = S_1, MAC_1, H, SN, sig(H, SN)$. Since S' is valid, by Lemma 1, there is $q \in [1, SN]$ such that $S_1^{(q)} = S_1^{(q)}, MAC_1^{(q)}, H, SN, sig(H, SN)$. Since sequence

number SN is only used in $A'^{(SN)}$, clearly the only value for q is SN . Also by Lemma 1, if $S'_1 \neq A'_1{}^{(SN)}$, then there is no polynomial algorithm that can generate a valid authenticated continuation of S' , given adaptively chosen streams, and any valid continuation of A' . So neither can E, even after obtaining $A'_2{}^{(SN)}$.

Inductive step: Suppose that the inductive claim is true for $j - 1$. We prove that it is also true for j . In order to do so, suppose B has received the valid stream $S' = (S'_1, \dots, S'_j)$. This means that it was possible to produce a valid continuation of (S'_1, \dots, S'_{j-1}) , and, by the inductive hypothesis, S_1, \dots, S_{j-1} must be authentic with probability $1 - \eta$, where η is negligible. We now have to prove that, if S_j is non-authentic, then E can cause B to receive a valid continuation of S' only with negligible probability. By Lemma 2, before B's receipt of S'_j , A has released at most j blocks during session SN . Therefore, the information available to E before B's receipt of S'_j consists of:

- the previous authenticated streams $A'^{(1)}, \dots, A'^{(SN-1)}$ sent by A, and
- the stream $A'^{(SN)} = A'_1{}^{(SN)}, \dots, A'_j{}^{(SN)}$ sent by A during session SN .

Let $S'_1 = S_1, MAC_1, H, SN, sig(H, SN)$. Since S' is valid, by Lemma 1, there is $q \in [1, SN]$ such that $S'_1{}^{(q)} = S_1^{(q)}, MAC_1^{(q)}, H, SN, sig(H, SN)$. Since sequence number SN is only used in $A'^{(SN)}$, clearly the only value for q is SN . Also by Lemma 1, if $S'_j \neq A'_j{}^{(SN)}$, then there is no polynomial algorithm that can generate a valid authenticated continuation of S' , given adaptively chosen streams, and any valid continuation of $A'^{(SN)}$. So neither can E, even after obtaining $A'_{j+1}{}^{(SN)}$. QED.

References

1. R. Anderson, F. Bergadano, B. Crispo, J.H. Lee, C. Manifavas, R.M. Needham, "A New Family of Authentication Protocols", *Operating Systems Review*, 32(4):9-20, 1998.
2. N. Asokan, G. Tsudik and M. Waidner, "Server Supported Signatures", *Proc. 1996 Esorics*, Rome, Italy, September 1996 pp. 131-143.
3. A. Ballardie, "Scalable Multicast Key Distribution", *IETF - RFC 1949*, May 1996.
4. M. Bellare, R. Canetti and H. Krawczyk, "Pseudorandom Functions Revisited: The Cascade Construction and its Concrete Security", *Proc. 37th Symposium on the Foundations of Computer Science*, IEEE, 1996.
5. F. Bergadano and B. Crispo, "Multiparty Authentication, Fast and Short", Seminar held at IBM T. J. Watson Research Center, August 1998. <http://www.research.ibm.com/security/seminar.html>
6. F. Bergadano, D. Cavagnino, B. Crispo, "Individual Single Source Authentication on the MBone", IEEE International Conference on Multimedia and Expo, New York, 2000.
7. D. Bleichenbacher, U. Maurer, "On the Efficiency of One-Time Digital Signatures", in *Advances in Cryptology — AsiaCrypt 96* (Springer LNCS).
8. R. Canetti and B. Pinkas, "A Taxonomy of Multicast Security Issues", *Internet Draft*, May 1998.

9. S. Cheung, "An Efficient Message Authentication Scheme for Link State Routing", *Proc. of 13th Annual Computer Security Applications Conference*, San Diego, California, 1997.
10. M. Dyer, T. Fenner, A. Frieze, A. Thomason, "On key storage in secure networks", *Journal of Cryptology*, vol. 8, 1995, pp. 189-200.
11. S. Even, O. Goldreich, S. Micali, "On-line / off-line digital signatures", in *Advances in Cryptology — CRYPTO 89* (Springer LNCS v. 435) pp. 263-275.
12. R. Gennaro, P. Rohatgi, "How to Sign Digital Streams", in *Advances in Cryptology — CRYPTO 97*, Springer LNCS v. 1294 pp. 180-197.
13. S. Goldwasser, S. Micali, R.L. Rivest, "A Digital Signature Scheme Secure Against Adaptive Chosen Message Attacks", in *SIAM Journal of Computing* v. 17 no. 2 (April 1988) pp. 281-308.
14. O. Goldreich, S. Goldwasser, S. Micali, "How to Construct Random Functions", *Journal of the ACM*, vol. 33:4, 1986, pp. 210-217.
15. N. M. Haller, "The S/key(tm) One-time Password System", *Proc. 1994 ISOC Symposium on Network and Distributed Security*, San Diego, CA, February 1997.
16. H. Krawczyk, M. Bellare and R. Canetti, "HMAC: Keyed-Hashing for Message Authentication", *IETF - RFC 2104*, February 1997.
17. L. Lamport, "Password Authentication with Insecure Communication", *Communication of the ACM*, 24:11, Nov. 1981, pp. 770-772.
18. R.C. Merkle, "A Digital Signature Based on a Conventional Encryption Function", in *Advances in Cryptology — CRYPTO 87* (Springer LNCS v. 293) pp. 369-378.
19. C. Metz, "Reliable Multicast: When Many Must Absolutely Positively Receive It", *IEEE Internet Computing*, pp. 9-13, July 1998.
20. S. Micali, "Enhanced Certificate Revocation System", *Technical Report MIT/LCS/TM-542* (November, 1985).
21. S. Mittra, "Iolus: a Framework for Scalable Secure Multicast", in *ACM SIGCOMM*, Cannes, September 1997.
22. A. Perrig, R. Canetti, J. D. Tygar and D. Song, "Efficient Authentication and Signing of Multicast Streams over Lossy Channels", *IEEE Symposium on Security and Privacy*, Oakland, California, USA, 2000.
23. R. L. Rivest and A. Shamir, "PayWord and MicroMint: Two Simple Micropayment Schemes", *Proc. 1996 Security Protocols Workshop*, Cambridge, UK, April 1996, pp. 69-87.
24. S. Schecter, T. Parnell, A. Hartemink, "Anonymous authentication of membership in dynamic groups", *Proc. Workshop on Financial Cryptography*, 1999.

A Global PMI for Electronic Content Distribution

Carlisle Adams and Robert Zuccherato

Entrust Technologies, 750 Heron Road
Ottawa, Ontario, Canada K1V 1A7
{cadams,robert.zuccherato}@entrust.com

Abstract. The paper describes a novel application of a Privilege Management Infrastructure (PMI) to enforce copyright protection in electronic content distribution. The PMI is “global” in nature and thus permits customers to gain access to content on any appropriate device. The use of a PMI also allows delegation of access to content. A unique key encrypting scheme provides increased security over other methods of protecting electronic content.

1 Introduction

The distribution of electronic content via the Internet is becoming more and more common. Electronic content includes such objects as electronic documents (e.g., PDF, Word documents), music (e.g., MP3), video (e.g., MPEG), and games. There are presently problems with this method of distribution. The most significant of these is that once the content has been downloaded it can easily be copied and re-distributed. Thus, enforcing copyright protection is difficult, particularly if the customer wishes to download the content for use when off-line or for use on multiple machines/devices. In addition, it is difficult to provide customers fine-grained access to content (e.g., it is difficult to allow individual customers to buy one article from a magazine), to reliably identify customers, and to allow customers to further delegate access to content in a controlled manner, when required. This proposal attempts to address these problems by taking advantage of some current solutions for authentication using a Public Key Infrastructure (PKI) (see [3] for an overview of PKI) as well as some new ideas using attribute certificates [9].

In order to provide a concrete example, this paper will describe a sample application that distributes PDF versions of magazine articles using a PDF viewer. Generalizations to other forms of electronic content should be relatively straightforward.

While this solution, like most solutions for content distribution, does not prevent unauthorized distribution by determined and malicious legitimate customers (especially in a software environment), it does prevent the typical user from doing so, while still allowing online or off-line use, access from different devices, further delegation and fine-grained access control.

2 What Is a PMI?

Within a Public Key Infrastructure a public key is bound to a user's identity through the use of a certificate. For example, in an X.509 [6,7] based PKI, a Certification Authority (CA) will verify the identity of the user and that he/she actually has the private key associated with the claimed public key. If the test of identity and Proof-of-Possession pass, then the CA will place the public key along with the user's identity in an X.509 public key certificate and sign this certificate using its private key. When any other entity wishes to verify a signature of or encrypt data for the user, he/she first verifies the signature on the certificate using the CA's public key. If the signature verifies then the public key contained in the certificate can be used for the desired purpose. Thus, end entities need only trust the CA's public key, typically achieved through some out-of-band method [2], in order to validate the certificates of other entities in the PKI. If an end entity trusts a CA and has obtained the CA's public key in a way that guarantees its authenticity, the CA's public key is known as the CA's root key.

A Privilege Management Infrastructure (PMI) [7] is similar to a PKI, except that instead of using public-key certificates to bind a user's identity to a public key, an attribute certificate is used to bind an identity to certain rights or privileges. An Attribute Authority (AA) that wishes to grant a user certain privileges will codify the privileges (usually represented by an attribute-value pair) and place them in an attribute certificate with the user's identity. The AA then signs the attribute certificate using its private key. When the user wishes to use those privileges to gain access to a protected resource he/she presents the attribute certificate to the entity controlling access (the "gatekeeper"). The gatekeeper will then authenticate the user and verify the signature on the attribute certificate using the AA's root public key. The gatekeeper must have already established trust in the AA's public key (again, typically achieved through some out-of-band mechanism). If the signature verifies and the attribute certificate contains the required attribute, the user is allowed access to the protected resource. In our example, the PDF viewer will act as the gatekeeper.

In a PKI, a CA can certify the public key of another subordinate CA, thus allowing end-entities that trust one CA to validate certificates of the subscribers in another CA domain. Similarly, an AA can grant privileges to another AA, thus allowing gatekeepers who trust one AA to accept attribute certificates issued by another AA. The gatekeeper must now verify that the intermediate AA has, in fact, been delegated authority to grant this privilege by the trusted AA. This process is referred to as delegation in [9]. We will also adopt that terminology.

3 The Idea

The idea is that a root Attribute Authority for a large Privilege Management Infrastructure (PMI) would control access to individual pieces of electronic content. Each PDF viewer, for example, would have the root key for this PMI embedded within it and access to the document would not be granted unless a

valid attribute certificate for the customer within that PMI existed. In addition, each viewer would require an embedded CA root key for a PKI to authenticate users and also a master symmetric key for obtaining access to content.

Thus, the content creator would encrypt each piece of electronic content. For example, the magazine could make articles from each issue available for purchase. Customers would authenticate themselves to the magazine website and pay for and download the articles desired. When the user wished to read the downloaded articles, the viewer would first require authentication of the customer and, if a valid attribute certificate existed, decrypt and display the article.

The advantage of using this method of distributing electronic content instead of just encrypting the content for each customer using, for example, CMS [5] or PKCS #7 [10] is that if the content is encrypted directly for the customer, he/she can simply decrypt and distribute the pirated content. If the proposed method were used however, the PDF viewer, for example, would only decrypt the content upon authentication of the customer and could make the plaintext difficult to obtain (e.g. would not write the content to disk).

There is, however, at least one potential problem with this scheme (see Section 5.1 below). This proposal will only make it more difficult for most legitimate customers to illegally gain access to or copy and distribute electronic content. Determined individuals (i.e., those with the ability to analyze executable code or those with access to the internal workings and components of their computer, device, etc.) will still be able to do bad things. Unfortunately, it appears that this will always be a property of e-content distribution schemes since at some point the plaintext content must appear somewhere on the customer's machine. If someone has the ability to analyze how the plaintext was obtained or to gain direct access to the plaintext as it is being displayed, they will always be able to compromise the system.

4 The Architecture

This section describes the proposed Privilege Management Infrastructure as well as the accompanying Public Key Infrastructure for authentication.

4.1 The PMI

In this architecture, the root key of the PMI is embedded in the PDF viewer (or the appropriate viewer for the type of content). It is envisioned that this root could be the root of a global PMI similar to the PKI roots that exist in web browsers. The root Attribute Authority would then issue an attribute certificate to the PDF viewer manufacturer indicating that it was authorized to produce PDF documents to be displayed by the viewer and that this privilege could be delegated. In a similar way, the root Attribute Authority could issue attribute certificates to any manufacturer of electronic content viewers. The PDF viewer manufacturer would then issue an attribute certificate to the magazine publisher indicating that it was authorized to produce PDF documents (i.e., that it could

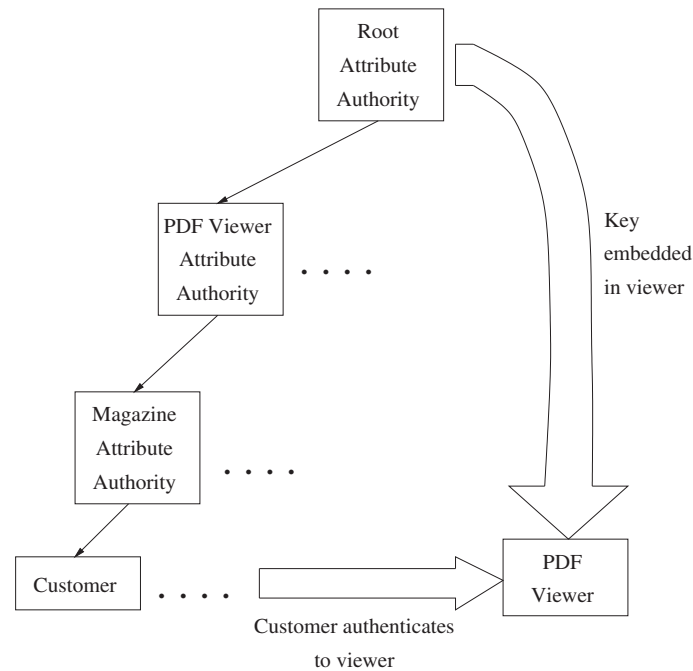


Fig. 1. The PMI architecture

authorize customers to have access to the encrypted content). The utility of having these layers of Attribute Authorities will be discussed further in Section 6.3. The need for a PMI, and in particular attribute certificates, will be described in Section 5.4.

Upon the purchase of an article, the customer would authenticate to the magazine publisher and provide it with a customer symmetric key. The magazine publisher would then issue an attribute certificate to the customer indicating that the customer was authorized to view the particular article and also including the content symmetric key used to encrypt the article, encrypted with the master symmetric key and then encrypted with the customer symmetric key. (The purpose of doubly encrypting the content symmetric key will be discussed further in Section 5.) The customer's attribute certificate could also place restrictions on when or how the content is to be viewed and may or may not allow further delegation. For example, a university library may subscribe to the magazine and then provide access to all of its students. The encrypted article including the customer's, the magazine publisher's and the PDF viewer manufacturer's attribute certificates would be delivered to the customer.

When the customer wishes to read the article, the viewer would authenticate and receive the customer symmetric key from the customer, and also verify the validity of the customer's, the magazine publisher's and the PDF viewer manufacturer's attribute certificates using its embedded PMI root key. If the

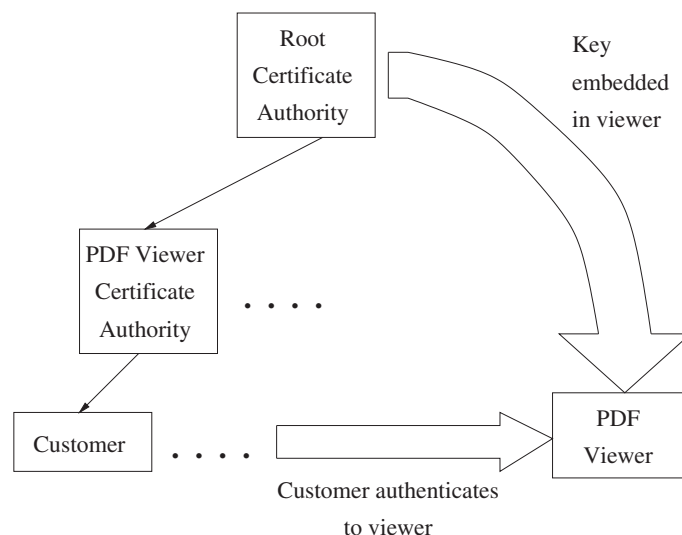


Fig. 2. The PKI architecture

complete attribute certificate chain validated, the customer symmetric key and the master symmetric key would be used to decrypt the content symmetric key that would be used to decrypt the article, which would be displayed to the user.

In this way, only legitimate customers that had paid for the article could view it.

4.2 The PKI

There are a number of possible PKI architectures that are compatible with the proposed PMI architecture. In fact any method of authenticating the customer could be used instead of a PKI. Here we describe one possible architecture.

Since the viewer must be able to authenticate legitimate customers, it must have the root of a PKI embedded within it. Then the root CA could certify the PDF viewer manufacturer's CA, which would in turn certify the customer (as well as, in this example, the magazine publisher).

4.3 How It Could Work

For example, when the customer downloads the PMI-enabled PDF viewer, he/she would also be enrolled in the PDF viewer manufacturer's PKI domain. (Alternatively, the customer could already belong to a PKI that could be chained to the PDF viewer manufacturer's PKI domain.) Then, when the customer wishes to purchase a magazine article he/she first authenticates him/herself to the magazine publisher using standard Internet authentication techniques (SSL/TLS [4] or SPKM [1], for example), provides it with the customer symmetric key over

the established session, pays for the content and obtains the content and appropriate attribute certificates. When he/she wishes to read the article, the PDF viewer would log the customer into their PKI identity (if he/she was not already logged in), authenticate the customer's identity (using, for example, techniques in ISO/IEC 9798-3 [8]) and obtain the customer symmetric key from the customer in order to determine whether or not to allow access to the content.

Thus, the sequence of events becomes:

1. Customer downloads the PDF viewer.
2. Customer enrolls in PDF viewer manufacturer's PKI. (If not already enrolled in a PKI.)
3. Customer authenticates to the magazine publisher.
4. Customer pays for the article and provides the magazine publisher with the customer symmetric key.
5. Magazine publisher encrypts the article with the content symmetric key. (Could be done in advance.)
6. Magazine publisher encrypts the content symmetric key with the master symmetric key to produce an encrypted symmetric key. (Could be done in advance.)
7. Magazine publisher encrypts the encrypted symmetric key with the customer symmetric key to produce a doubly encrypted symmetric key.
8. Magazine publisher Attribute Authority creates an attribute certificate for the customer containing the doubly encrypted symmetric key.
9. Magazine publisher sends the encrypted article and the attribute certificates to the customer.
10. Customer authenticates to the PDF viewer and provides the customer symmetric key.
11. The PDF viewer validates the attribute certificate.
12. The PDF viewer decrypts the content symmetric key using the customer symmetric key and the master symmetric key.
13. The PDF viewer decrypts the content using the content symmetric key.
14. The PDF viewer displays the content.

5 Security Issues

5.1 Encrypting the Content

Each piece of content would be encrypted with a unique content symmetric key. This encryption would only have to be performed once for each piece of content. The content symmetric key would then be encrypted with a master symmetric key and a customer symmetric key and placed within each customer's attribute certificate. The master symmetric key would be embedded in each viewer to allow decryption of the content symmetric key and thus, the content. This key should also be different for each type of viewer (i.e. for each type of e-content). The customer symmetric key, which would also be required to obtain access to the content, should be stored securely for the legitimate customer in such a way

that it is portable to different machines/devices. A simple solution is to store it within the customer's Personal Security Environment (PSE) [2]. A PSE is a software or token based secure storage for the customer's private keys and other sensitive information. Most PSEs can be moved from one device to another.

In this solution knowledge of both the master symmetric key and the customer symmetric key is required to gain access to the content. Thus, both must be securely protected. If the customer symmetric key is stored in the customer's PSE, it should only be available to the legitimate customer. The master symmetric key must be embedded in each viewer, however, and thus must not be readily available by analysis of the viewer executable. This could be accomplished by implementing a function whose sole purpose is to decrypt keys encrypted with this master symmetric key value. In other words, the plaintext key need not appear in memory and need not be passed into a general purpose decryption algorithm. A function could be used that would only perform decryption with the given key. The key then wouldn't need to actually appear in memory since bit operations could be used to optimize and obfuscate decryption with this code. Even so, the master symmetric key could become available to determined adversaries. However, unless they have the cooperation of a legitimate customer in order to acquire a customer symmetric key, they are no further ahead. Thus, determined legitimate customers may be able to get access to the plaintext, but this may not be preventable (see Section 5.2).

Note that if the viewer is implemented in secure hardware, then it is highly unlikely that the master symmetric key could be obtained and thus the solution described in this paper is very secure. For this reason this solution is more applicable to hardware implementations.

In addition, to support encryption the master symmetric key would have to be kept in a secure location so as not to be compromised. A Trusted Third Party (e.g. the root CA or root AA) could keep this key in secure hardware and encrypt content symmetric keys for content creators. The content creators (e.g. the magazine publisher) would authenticate themselves to the Trusted Third Party and present their attribute certificate indicating that they are legitimate creators. They could then provide the content symmetric key to the Trusted Third Party and it would be encrypted using the master symmetric key. Again, this operation would only need to be performed once for each piece of content. This encrypted key would then be encrypted again for each customer using the customer symmetric key.

The content, master and customer symmetric keys could, in fact, all be asymmetric keys. However, it is recommended that symmetric key cryptography be used for these keys, to allow for more efficient operations at the server.

5.2 Making Plaintext Unavailable

Using the other methods described in this document to restrict access to electronic content will not be successful if the decrypted content is somehow made available or stored on the user's disk, allowing copying of the content and unauthorized distribution. Thus, viewers should keep the plaintext in memory. How-

ever, even then, determined individuals could certainly read the content by scanning memory. Therefore, again, determined legitimate customers may be able to gain access to the plaintext content.

In some applications the disclosing of plaintext content may not be undesirable for certain customers. In these cases, the customer's attribute certificate could indicate whether or not the decrypted content should be made easily available to them.

5.3 Further Delegation

One of the advantages of this scheme is that it allows further delegation of privilege to access the electronic content. Let us consider the example of a university library that wishes to grant access to magazine articles to its students. The library has an attribute certificate containing its unique identifier, an indication of the privilege to view the content, and the content symmetric key encrypted with the master symmetric key and also the library's customer symmetric key. In order to delegate access, the attribute certificate must also contain an indication that the library is in fact allowed to delegate access.

When it does wish to delegate access to the magazine articles, the library will create an attribute certificate for each student to which access will be granted. The new attribute certificates will contain an identifier for the student to which access is granted, an indication of the privilege to view the content and the content symmetric key. The content symmetric key will be encrypted with the master symmetric key and the student's customer symmetric key. The library can produce this encrypted key by taking the doubly encrypted key out of its attribute certificate, decrypting it using its own customer symmetric key (leaving the singly encrypted key) and then encrypting it with the student's customer symmetric key.

The library must obtain the student's customer symmetric key in order to place the properly encrypted content symmetric key in the attribute certificate. Thus, it may make sense in these circumstances (and, in fact, any situation where the AA cannot be trusted with the customer symmetric key) for the customer to produce different symmetric keys for each application.

5.4 Why Attribute Certificates?

One may be tempted to not use attribute certificates at all in this type of scheme. Shouldn't the presence of the content symmetric key encrypted with both the master and customer symmetric keys be enough evidence that the customer had been granted access to the e-content?

Unfortunately, this is not the case. Since the outer encrypting of the content symmetric key is performed using the customer symmetric key, a malicious customer could very easily remove this encryption and encrypt it with any other symmetric key, thus easily delegating access. This would be undesirable. Attribute certificates eliminate this security weakness by placing this doubly encrypted key inside a signed object that cannot be created by the customer.

Note that it is not feasible in a large scale environment to reverse the order of encrypting so that the outer encrypting is performed by the master symmetric key. This change would require that the content symmetric key must be encrypted with both the customer and master symmetric key each time a customer was granted access. This would mean that the master symmetric key must be kept on-line which will decrease efficiency and could make the key vulnerable to attackers which attempt to break into the server in which it resides. With the present scheme, the content symmetric key need only be encrypted with the master symmetric key once, and then encrypted with a customer symmetric key each time a customer is granted access.

6 Other Issues

6.1 Anonymous DNs

A PKI could be used for authentication of customers. However, it is possible that some customers would not want their name or other vital information to appear in a widely available certificate. For such environments, it is recommended that anonymous DNs be used. In this example, the PDF viewer manufacturer's CA may be required to keep a database linking the anonymous DNs with actual identity information. Also, naming rules must be enforced so that each customer receives a unique DN within this PKI. It may also be desirable for customers to have different certificates (and DNs) for each viewer for which he/she is registered.

6.2 Certificate Rollover

In many cases it would be undesirable if a customer bought a song and after 6 months he/she couldn't use it because his/her public key certificate had expired. There are two possible solutions to this problem. One solution is to make customer certificates very long-lived (e.g. 10, 20 years). However, issuing long-lived public key certificates to end entities is discouraged in most environments for security reasons.

A second solution is to make the key short-lived (e.g. 6 months or a year) and require that every few months users must re-connect to the Internet to contact the PDF viewer manufacturer's CA and obtain a new public key certificate. A warning would have to be displayed when certificate expiry is approaching which advises customers of this requirement. This has the disadvantage that people who remain off-line for extended periods of time lose access to all of their electronic content. In order to link the attribute certificate issued to the customer with any public key certificate issued to that customer by the PDF viewer manufacturer's CA, the customer should be identified by their DN in the attribute certificate.

When the public key certificate of an attribute authority expires, however, all attributes issued by that authority can no longer be verified. Thus, attribute authorities must have keys that are very long lived (e.g. 20 years).

6.3 Are Global Root Keys Required?

The description in this paper assumed for simplicity that there would be one global root key for the PMI that would be used for all types of electronic content, one global root key for the PKI that would identify each customer of electronic content, and one master symmetric key for decrypting content. This is not strictly necessary. The PDF viewer manufacturer, for example, could establish its own roots for the PKI and PMI and master symmetric key that would be embedded within each PDF viewer. In some situations, this configuration may be more desirable.

7 A Comparison with Other Schemes

This section will describe other possible solutions for distributing electronic content and compare them with the solution proposed in this paper.

7.1 Encrypting the Content Just for the Customer

Another method of allowing the secure downloading of electronic content so that it is only accessible by the legitimate customer is to simply encrypt the content for the customer. The customer simply generates a (symmetric or asymmetric) key and sends it to the e-content distributor who encrypts the content for the user. This solution is conceptually simple and also allows the user to gain access to the content on different devices. However, it is now very easy for malicious customers to decrypt the content and distribute the plaintext. While it is also possible with the scheme described in this paper for malicious customers to gain access to plaintext by gaining access to the master symmetric key, it is much more difficult than simply performing a decryption using a key known to the customer.

Similarly, it is possible for a malicious customer to sell his/her PSE and password, thus allowing others to obtain access to all content he/she has purchased. Customers will be deterred from doing this for two reasons. First, any one with access to the customer's PSE would also be able to impersonate the customer, thus potentially incurring a large amount of costs for the customer. Secondly, if unauthorized redistribution of electronic content occurs on a large scale, the presence of the customer's PSE among a large number of people allows authorities to trace the source back to the malicious customer.

Instead of encrypting the content directly for the customer, an alternative solution is to encrypt it for the customer's computer. A (symmetric or asymmetric) key could be generated on the customer's computer and stored in such a way that it is only accessible on that computer. For example, it could be encrypted by a key generated from unique data on the host computer. This makes it difficult for malicious customers to gain access to plaintext, but does not allow customers to view/play the content on different computers or devices.

In addition, neither of these solutions allows secure delegation of access.

7.2 Encrypting the Content Just for the Viewer

It may also be tempting to encrypt the content using just a key that is embedded in the viewer. This solution allows anyone with a copy of the viewer to have access to the content, however. This may make sense in situations where sales of the viewer are projected to be more important than sales of the content, but that business model is seldom the one envisioned in current and projected e-content distribution ventures.

This solution also suffers from the problem that if someone is able to find the decryption key in the viewer and distribute it, unlimited access to all content for everyone may be available.

While the solution described in this paper also relies upon a key embedded in the viewer, loss of this key does not immediately provide unlimited access to all content. Only a legitimate customer can obtain access. Thus, this solution provides additional security over simply encrypting content for the viewer, and also allows a more realistic business model.

8 Conclusion

This paper described a method for enforcing copyright protection on a per-customer basis. The described solution allows both online and off-line use, provides customers access to content on any device that has the appropriate viewer, allows further delegation of access, and is secure except against very determined malicious legitimate customers.

References

1. C. Adams, "The Simple Public-Key GSS-API Mechanism (SPKM)", RFC 2025, October 1996.
2. C. Adams and S. Farrell, "Internet X.509 Public Key Infrastructure Certificate Management Protocols", RFC 2510, March 1999.
3. C. Adams and S. Lloyd, *Understanding Public-Key Infrastructure; Concepts, Standards, Deployment Considerations*, Macmillan Technical Publishing, 1999.
4. T. Dierks and C. Allen, "The TLS Protocol Version 1.0", RFC 2246, January 1999.
5. Housley, R., "Cryptographic Message Syntax", RFC 2630, June 1999.
6. Housley, R., Ford, W., Polk, W. and D. Solo, "Internet X.509 Public Key Infrastructure: Certificate and CRL Profile", RFC 2459, January 1999.
7. "Information Technology – Open Systems Interconnections – The Directory: Authentication Framework", ISO/IEC International Standard 9594-8 — ITU-T Recommendation X.509 (1997).
8. "Information Technology – Security Techniques – Entity authentication - Part 3: Mechanisms using asymmetric signature techniques", ISO/IEC International Standard 9798-3: 1998. (2nd edition)
9. ITU-T Recommendation X.509 (1997) – ISO/IEC 9594-1:1997, Information Technology – Open Systems Interconnections – The Directory: Authentication Framework — Draft Amendment (DAM) 1: Draft Amendment on Certificate Extensions.
10. Kaliski, B., "PKCS #7: Cryptographic Message Syntax, Version 1.5.", RFC 2315, March 1998.

A Polynomial-Time Universal Security Amplifier in the Class of Block Ciphers

John O. Pliam

Laboratory for Security and Cryptography (LASEC),
Swiss Federal Institute of Technology, Lausanne (EPFL),
LASEC-DSC-EPFL, CH-1015 Lausanne, Switzerland
`john.pliam@epfl.ch`

Abstract. We demonstrate the existence of an efficient block cipher with the property that whenever it is composed with any non-perfect cipher, the resulting product is strictly more secure, against an ideal adversary, than the original cipher. We call this property universal security amplification, and note that it holds trivially for a one-time pad (a stream cipher). However, as far as we are aware, this is the first efficient block cipher with this property. Several practical implications of this result are considered.

1 Introduction

It is often asked in cryptography whether the product of two ciphers might be more or less secure than one of the ciphers by itself. An *amplification* of security doesn't happen in general and important counterexamples have been identified. For example, if the permutations of a block cipher form a group (or more precisely, are uniformly distributed on a subgroup of the symmetric group on the set of message blocks), then two-key double encryption is no better than single encryption. Thus, it has been seen as important to rule out this pathology in the case of DES [4]. Furthermore, the security of a product can actually be less than that of the second cipher when the plaintext statistics are ill-behaved with respect to the permutations of the first cipher [14]. Nevertheless, depending on how security is measured and how the ciphers are modeled, other affirmative results have been advanced [20,8,1].

In this paper, we take a novel approach to this problem, raising a strong existence question about the security of product ciphers. Specifically, we ask: Is there an efficient block cipher which amplifies the security, against an ideal adversary, of every non-perfect cipher with which it is composed? By construction, we answer this question in the affirmative.

The constructed cipher, as presented, would not be widely viewed as practical because it requires a variable length key which grows with the amount of plaintext encrypted (much like a one-time pad). On the other hand, if a cryptographically strong substitute for the key were used (such as a key schedule, hash, or pseudo-random function), then the strength of the security amplification would be no worse than the strength of the key substitute.

There are other practical implications of our result. First of all, the techniques used here could facilitate the construction of computationally efficient primitives (such as dynamic S-boxes) with provably strong security properties. More generally, if we are to understand, in more than purely heuristic terms, the security convergence of modern iterated cryptosystems, then our result establishes new limits on what can be accomplished in polynomial-time. Our construction might be modified and compromised to obtain faster ciphers with complementary security results.

2 Preliminaries

A basic familiarity with random variables and probability spaces [11] is assumed. Some group theory [19] is also assumed, but in the next subsection, we shall review some important terminology about permutation groups [7].

2.1 Permutation Groups

Let \mathcal{X} be any set. The collection of all invertible functions on \mathcal{X} forms the *symmetric group* $\mathfrak{S}_{\mathcal{X}}$. Any subgroup $G \leq \mathfrak{S}_{\mathcal{X}}$ is called a *permutation group*, and we also say that G *acts on* \mathcal{X} and that \mathcal{X} is a G -*set*. The subgroup of G which fixes a point $x \in \mathcal{X}$ is called the (*point*) *stabilizer* of x , and is given by $\text{Stab}_G(x) = \{h \in G \mid hx = x\}$.

When studying n -bit block ciphers, the finite set $\mathcal{M} = \{0, 1\}^n$ of all n -bit binary strings (or equivalently the integers $\{0, 1, \dots, 2^n - 1\}$) is the most natural G -set for some permutation group $G \leq \mathfrak{S}_{\mathcal{M}}$. But additionally for this paper, we will often consider two other actions of G on related sets. By $\mathcal{M}^{(\ell)}$ we mean the set of tuples of size ℓ with distinct elements in \mathcal{M} , G acting elementwise. If $p = (p_1, \dots, p_{\ell}) \in \mathcal{M}^{(\ell)}$, the point stabilizer $\text{Stab}_G(p)$ is sometimes written $\text{Stab}_G(p_1, \dots, p_{\ell})$. By $\mathcal{M}^{\{m\}}$ we mean the set of subsets of \mathcal{M} of size m , where $g \in G$ acts on $S \in \mathcal{M}^{\{m\}}$ by taking $S \mapsto gS$. The point stabilizer of $S \in \mathcal{M}^{\{m\}}$ is sometimes written $\text{Stab}_G\{S\}$.

2.2 Shannon's Model and Product Ciphers

Following Shannon [20], we model an n -bit *block cipher* as a $\mathfrak{S}_{\mathcal{M}}$ -valued random variable. If a cipher X only takes values in a subgroup $G \leq \mathfrak{S}_{\mathcal{M}}$, then X may be called a G -*cipher*. We may model a stream cipher in the same spirit (cf. [15]). Let $\{0, 1\}^*$ denote the (infinite) set of finite binary strings, and let $H \leq \mathfrak{S}_{\{0, 1\}^*}$ be the subgroup of length-preserving permutations. We shall call an H -valued random variable a *stream cipher*¹. By a *cipher* we mean either a block cipher or a stream cipher.

¹ In practice, a stream cipher will typically also have consistent *block prefix action*, i.e. for some integer n , it will be confined to permutations $h \in H$ such that when $|u| = |u'| \in n\mathbb{Z}$, $h(uw) = u'w'$ implies that for all v of length $|w|$, $h(uv) = u'v'$ for some v' .

Given two independent ciphers X and Y acting on the same message space, the cipher XY is called a *product cipher*, Y is called its *first component* and X is called its *second component*. The distribution of the product of two block ciphers is given by the *convolution*,

$$\mathbf{P}[XY = g] = x * y(g) \triangleq \sum_{h \in G} x(gh^{-1})y(h), \quad (1)$$

where $x(g) = \mathbf{P}[X = g]$ and $y(g) = \mathbf{P}[Y = g]$. This representation of a product cipher will prove useful in the sequel.

The cipher U which is uniformly distributed on $\mathfrak{S}_{\mathcal{M}}$ is called the *perfect cipher*. For any subgroup $G \leq \mathfrak{S}_{\mathcal{M}}$ the G -cipher U_G which is uniformly distributed on G is called the *uniform G -cipher*. Given an infinite sequence of independent and uniformly random bits, z_0, z_1, \dots , we may form a simple stream cipher, called the *one-time pad*, by mapping plaintext word m into $z_{|m|} \oplus m$, where $z_{|m|}$ is the word $z_0 \cdots z_{|m|}$.

2.3 The Computational Model

Shannon's model is a purely probabilistic one; it says very little about how a computer might transform plaintext into ciphertext and back. For a cipher X to be practical, there should be effective procedures for encryption (computing the action of X on plaintext) and decryption (computing the action of X^{-1} on ciphertext).

One natural choice for the computational model is the standard *Turing machine* model [9]. Informally, we have an encryption algorithm Enc , which has as input arguments the plaintext m and the random key k , and which outputs ciphertext c . The corresponding decryption algorithm Dec is similarly defined. Formally in this model, we require a pair of deterministic Turing machines E and D , such that (under suitable encoding) $m = D(k, E(k, m))$, for all m and k . Notice that under this model, all randomness enters as an argument to the encryption and decryption algorithms, or equivalently as input data on the Turing machine tapes. Our view is that this model of computation is unnecessarily restrictive, because it fails to capture the simple idea that some ciphers (like the one-time pad) are “computationally efficient” even though they may require impractical amounts of key material to encrypt *every* possible plaintext.

Alternatively, we consider encryption and decryption algorithms which access key material as an auxiliary subroutine call. Formally, such a subroutine call is idealized by an *oracle function* $f : \{0, 1\}^* \rightarrow \{0, 1\}$, and we are thus invoking the computational model of an *oracle Turing machine (OTM)* [9]. An OTM is a deterministic Turing machine augmented by an *oracle tape* and additional logic so that that at any time, the oracle tape with input α written on it can, in one step of computation, be transformed to have $f(\alpha)$ written on it. An OTM M with specific oracle function f will be denoted by M^f , and its time complexity is computed in the usual way (with oracle evaluation counting as one step). We may model uncertainty about the oracle function by treating it as an instance of a *random oracle function* $F : \{0, 1\}^* \rightarrow \{0, 1\}$.

The next two definitions capture our intuitive notion of efficient encryption/decryption for block and stream ciphers, respectively.

Definition 1 (Efficient Block Ciphers). *An ensemble of block ciphers $\{X_n\}$, $n \in \mathbb{N}$ will be called **computable in polynomial-time** if there exists a random oracle function F and a pair of polynomial-time OTM's, E and D , such that for each $n \in \mathbb{N}$: (i). for each $p \in \{0, 1\}^n$, $p = D^F(E^F(p))$, and (ii). the distribution of E^F , restricted to strings of length n , is identical to that of X_n , and (iii). the distribution of D^F , restricted to strings of length n , is identical to that of X_n^{-1} .*

By a mild but common abuse of notation, a block cipher X acting on $\{0, 1\}^n$ will be called *computable in polynomial-time* if it is one of an ensemble of such ciphers, and any important properties hold for each representative.

Definition 2 (Efficient Stream Ciphers). *A stream cipher X will be called **computable in polynomial-time** if there exists a random oracle function F and a pair of polynomial-time OTM's, E and D , such that: (i). for each $p \in \{0, 1\}^*$, $p = D^F(E^F(p))$, and (ii). the distribution of E^F is identical to that of X , and (iii). the distribution of D^F is identical that of X^{-1} .*

Note that by Definitions 1 and 2, both the one-time pad and the Luby-Rackoff construction [15] are efficient. In fact, each is computable in linear time. Notice also that being computable in polynomial-time does not preclude that exponentially many bits may be necessary to completely describe the cipher's action on the entire message space. For example, each round of the Luby-Rackoff construction (a Feistel cipher with a perfectly random function acting on half-words) takes on one of

$$\left(2^{\frac{n}{2}}\right)^{2^{\frac{n}{2}}}$$

distinct permutations of an n -bit message space. Thus, for the common 3-round version of the construction, there must be $3n2^{\left(\frac{n-2}{2}\right)}$ bits to entirely describe it.

However, neither the one-time pad nor the Luby-Rackoff construction meets our objective. The one-time pad is not a block cipher. Furthermore, every permutation of the Luby-Rackoff construction is even and hence is confined to a proper subgroup (the alternating group, $\mathfrak{A}_{\mathcal{M}} \leq \mathfrak{S}_{\mathcal{M}}$), and we shall see from Lemma 1 below that it cannot be a universal security amplifier.

2.4 Optimal Chosen Plaintext Attacks

We now introduce the measure of security in terms of which strict security inequalities will be derived. Informally, it is just the average cost of the optimal (non-adaptive) chosen plaintext attack for an adversary in possession of an oracle which will answer the question, “is $X = g$?”. There are two stages to the optimal strategy. First the adversary discards all permutations which are inconsistent with the acquired plaintext-ciphertext pairs. Then among the remaining permutations, he queries the oracle for the exact permutation in order of non-increasing probability. The adversary will obviously choose the plaintexts such

that the average cost of this strategy is minimized. The difficulty of this attack is a direct and meaningful measure of the cipher's security.

To formally quantify this attack against a G -cipher X , $G \leq \mathfrak{S}_{\mathcal{M}}$, let us assume that the adversary has collected ℓ plaintexts and their corresponding ciphertexts into tuples $p, c \in \mathcal{M}^{(\ell)}$, respectively. The ciphertext tuple c is an instance of the random variable $C^\ell = Xp$, whose uncertainty is due exclusively to uncertainty about X . Now for any random variable Z the average cost of guessing its value is called the *guesswork*² of Z and is given by

$$W(Z) \triangleq \sum_{i=1}^m p_{[i]} i, \quad (2)$$

where Z takes on m values, and where the probabilities of Z have been arranged according to $p_{[i]} \geq p_{[j]}$ for all $i < j$. For fixed p and c , the *conditional guesswork* $W(X|c, p)$ is the guesswork of X as in Equation (2) after discarding all permutations $g \in G$ such that $c \neq gp$, and then rearranging and rescaling the probabilities accordingly. Now we must still account for the uncertainty about C^ℓ . Evidently, for a particular choice of plaintext tuple p , the cost of the attack must be weighted by the *a posteriori* probabilities $\omega(c|p) = \mathbf{P}[C^\ell = c | p]$, yielding

$$W(X|C^\ell, p) = \sum_{c \in \mathcal{M}^{(\ell)}} W(X|c, p) \omega(c|p). \quad (3)$$

The minimum value of $W(X|C^\ell, p)$ is the *optimal chosen plaintext attack work factor*, which will be denoted

$$\theta_\ell(X) = \min_{p \in \mathcal{M}^{(\ell)}} W(X|C^\ell, p). \quad (4)$$

For continuity we take $\theta_0(X)$ to be $W(X)$.

3 The Main Result

3.1 The Existence Theorem

We shall prove by construction the following theorem.

Theorem 1. *There is a cipher X , computable in polynomial-time, such that for each $0 \leq \ell \leq 2^n$ and every independent cipher Y , $\theta_\ell(XY) \geq \theta_\ell(Y)$. Furthermore, equality holds iff $\theta_\ell(Y) = \theta_\ell(U)$.*

It is easily seen (see e.g. [17]) that no non-perfect cipher Y can have $\theta_\ell(Y) = \theta_\ell(U)$, for all ℓ . Thus this theorem tells us in a very meaningful way, that every non-perfect cipher is brought closer to the the perfect cipher by left multiplication by X .

The proof of Theorem 1 relies on three lemmas which treat different aspects of the problem. To express these lemmas succinctly, we introduce some additional

² Guesswork has sometimes been called *guessing entropy*, cf. [18] and [3]

terminology. First, the *support* of a G -cipher (or indeed any random variable) may be defined as $\text{supp}(X) \triangleq \{g \in G \mid \mathbf{P}[X = g] \neq 0\}$. Second, it is useful to denote the size of the smallest ℓ -message stabilizer of a group by $M_G(\ell) \triangleq \min_{p \in \mathcal{M}^{(\ell)}} |\text{Stab}_G(p)|$. It is easily seen that $\theta_\ell(U_G) = \frac{1}{2}[1 + M_G(\ell)]$.

The first lemma from [17] treats the case $\ell = 0$ but is also useful in establishing the other results.

Lemma 1. *Given $G \leq \mathfrak{S}_{\mathcal{M}}$, let X be a G -cipher. Every independent non-uniform G -cipher Y satisfies $W(XY) > W(Y)$, iff for each $g \in G$ and each subgroup $H \neq G$, $\text{supp}(X) \not\subseteq gH$.*

The next lemma provides sufficient conditions for nearly universal amplification ($\ell > 0$) for ciphers in any permutation group.

Lemma 2. *For a permutation group $G \leq \mathfrak{S}_{\mathcal{M}}$, let X be a G -cipher such that $\text{supp}(X) = G$. Then for each $1 \leq \ell \leq 2^n$ and every independent G -cipher Y , $\theta_\ell(XY) \geq \theta_\ell(Y)$. Furthermore, equality holds iff $\theta_\ell(Y) = \theta_\ell(U_G)$.*

The final lemma asserts the existence of a cipher suitable to translate Lemmas 1 and 2 into Theorem 1.

Lemma 3. *There is a cipher X , computable in polynomial-time, with $\text{supp}(X) = \mathfrak{S}_{\mathcal{M}}$.*

Assuming the validity of the above lemmas, the proof of Theorem 1 is immediate.

The proof of Lemma 2 is rather involved and is sketched in Sect. 4.4. Most of the rest of this paper is devoted to the construction of X and the proof of Lemma 3. Before diving into the precise details in Sect. 4, let us first take a slightly more informal look at the ideas underlying this construction.

3.2 An Intuitive Glimpse at the Construction

The symmetric group on the message space is truly enormous. Its size is approximated by

$$\log \log(2^n!) \approx n + \log(n) = O(n).$$

Because it takes two logarithms to bring $2^n!$ down to the polynomial n , our construction will exhibit two distinct sources of algorithmic efficiency:

1. *Recursion:* The cipher X will be recursively defined as the product of simpler ciphers. More precisely, the encryption algorithm **Enc** will itself be recursive but will also call another recursive algorithm **invSort**. The decryption algorithm **Dec** will be similarly defined. The time complexity and recursion depth of each algorithm will be a polynomial in n .
2. *Oblivious Action*³: The cipher X will be representable as the product of a large number of random powers of transpositions (i.e. permutations of message blocks two at a time). Then **Enc** and **Dec**, the defining algorithms of X , will make use of only polynomially many transpositions for every block encrypted.

³ We borrow this term from [16] where it is used in the same context

Let $G \leq \mathfrak{S}_{\mathcal{M}}$ be any permutation group. There are many ways to construct a product cipher PQ which achieves every permutation in G , even though *both* P and Q are sparse on G . Indeed for any subgroup $H \leq G$, we may take Q which achieves every permutation in H and P which achieves one permutation in every left coset of H in G . It is easy to see that PQ achieves every permutation in G . For many large groups, it is possible to find subgroups satisfying $|G| \gg |H|$ and $|G| \gg [G : H]$. Formally, we have an amplification of support: $|\text{supp}(PQ)| \gg |\text{supp}(P)|$, and $|\text{supp}(PQ)| \gg |\text{supp}(Q)|$. Thus by exploiting the algebraic structure of the group, we may construct a densely distributed cipher as a product of very sparsely distributed ciphers.

Let's try to carry this idea even further. Consider a chain of subgroups of G

$$\{1\} = H_0 \leq H_1 \leq \cdots \leq H_m = G,$$

and for each i , an H_i -cipher P_i which contains one permutation in every left coset of H_{i-1} in H_i . Then by simple induction, the product cipher $P_m \cdots P_2 P_1$, would have complete support on G . For example in the symmetric group on 2^n symbols, consider the subgroups $H_i = \text{Stab}(1, \dots, 2^n - i)$, $0 \leq i \leq 2^n$. On the one hand, this choice of subgroups is promising because the number of cosets in \mathfrak{S}_{2^n} of the largest proper subgroup is the polynomial n . Unfortunately however, there are 2^n subgroups in this chain, and so the number of terms in the product $P_m \cdots P_2 P_1$ grows exponentially with n . If we are to employ this technique, it may be inconvenient to use a chain of subgroups which fix collections of words in \mathcal{M} – either as tuples or as sets – because any hierarchy of such collections would typically be as large as \mathcal{M} itself.

It thus makes more sense to define subgroups which fix some feature of the words in \mathcal{M} . To that end define K_i to be the subgroup consisting of the permutations of $\mathfrak{S}_{\mathcal{M}}$ which preserve the first $n - i$ bits of each message block. We shall call K_i the $(n - i)$ -bit *prefix stabilizer subgroup* of $\mathfrak{S}_{\mathcal{M}}$, and as i ranges from 0 to n these form the chain of subgroups

$$\{1\} = K_0 \leq K_1 \leq \cdots \leq K_n = \mathfrak{S}_{\mathcal{M}}. \quad (5)$$

We will construct, for each $1 \leq i \leq n$, a K_i -cipher P_i which contains one permutation in every left coset of K_{i-1} in K_i . Then the cipher of Lemma 3 will be defined as

$$X = P_n \cdots P_2 P_1. \quad (6)$$

But let us compute the minimal support required of P_n . That is to say let us count the number of left cosets of K_{n-1} in $\mathfrak{S}_{\mathcal{M}}$. Since K_{n-1} permutes all but the most significant bit of words in \mathcal{M} , the left cosets of K_{n-1} are characterized by the rearrangements of \mathcal{M} with distinct patterns of the most significant bit. There are precisely

$$[\mathfrak{S}_{\mathcal{M}} : K_{n-1}] = \binom{2^n}{2^{n-1}}$$

of these rearrangements. Observe that while we have reduced the number of permutations by a large number (by $(2^{n-1}!)^2$ in fact), on a doubly logarithmic scale we still have

Table 1. A randomly chosen arrangement of $\{0, 1, \dots, 7\}$ is sorted with respect to the most significant bit after the application of only 3 rounds of disjoint transpositions. The first two columns indicate the initial arrangement (position, value). The next three columns give, for each round, the transposition affecting the value at that position and subsequent arrangement

| | | round (transp./arrangement.) | | | |
|---|---------|------------------------------|----------|------|-----|
| | | 0 | 1 | 2 | |
| 7 | 2 = 010 | (67) 100 | () 100 | () | 100 |
| 6 | 4 = 100 | (67) 010 | (46) 111 | () | 111 |
| 5 | 1 = 001 | () 001 | () 001 | (15) | 101 |
| 4 | 7 = 111 | () 111 | (46) 010 | (04) | 110 |
| 3 | 0 = 000 | (23) 101 | (13) 011 | () | 011 |
| 2 | 5 = 101 | (23) 000 | () 000 | () | 000 |
| 1 | 3 = 011 | () 011 | (13) 101 | (15) | 001 |
| 0 | 6 = 110 | () 110 | () 110 | (04) | 010 |

$$\log \log \left(\frac{2^n}{2^{n-1}} \right) \approx n + 1 = O(n).$$

It may appear that we are right back where we started, yet we have transported the problem onto very fertile new ground.

The efficiency in our algorithms for P_i has its heritage in the closely related problem of card shuffling. In fact, both the security of a product cipher [17] and the fairness of a shuffled deck of cards [2,6] is related to the uniformity of convolutions as in (1). In their now famous analysis riffle shuffles, Aldous and Diaconis remarked that “the lovely new idea here is to consider shuffling as inverse sorting.” [2, Remark (a), p. 344]. Indeed it is quite natural to consider *encryption* as inverse sorting because the rearrangements of \mathcal{M} which characterize the left cosets of $K_{n-1} \leq \mathfrak{S}_{\mathcal{M}}$ correspond precisely with the permutations which would be used in the *first* step of the obvious recursive sorting algorithm. In the reverse order, we may achieve all permutations of \mathcal{M} by first achieving all rearrangements of the most significant bit, and then proceeding recursively with the less significant bits. What we claim is that sorting and inverse sorting on the most significant bit can be done in polynomial-time using both recursion and the oblivious action of transpositions. The rest is gravy.

Let us demonstrate this efficiency in a simple example with $n = 3$ and thus $\mathcal{M} = \{0, 1, \dots, 7\}$. We start with a random arrangement (6, 3, 5, 0, 7, 1, 4, 2) of the elements of \mathcal{M} , and attempt to sort this tuple on the most significant bit by the application of $n = 3$ rounds of involutions (recall that every involution is a product of disjoint transpositions). For reasons of efficiency we shall restrict ourselves to transpositions of the form $(j, j \oplus 2^i)$, with i constant for every round. The allowable round involutions are $(01)^{b_1}(23)^{b_2}(45)^{b_3}(67)^{b_4}$, $(02)^{b_5}(13)^{b_6}(46)^{b_7}(57)^{b_8}$ and $(04)^{b_9}(15)^{b_{10}}(26)^{b_{11}}(37)^{b_{12}}$, for rounds 0, 1 and 2, respectively. Table 1 below shows that we can indeed sort on the most significant bit of 2^n integers by carefully choosing the powers b_i in only only n rounds.

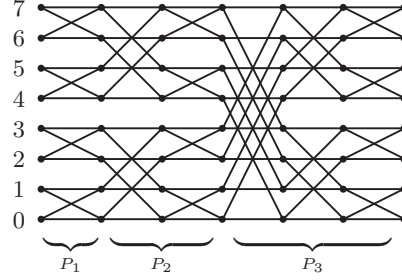


Fig. 1. The structure of the cipher $X = P_3P_2P_1$ for $n = 3$. The rounds are applied left to right, and each round corresponds to a random involution shown as a vertical column of butterflies. Each butterfly in the diagram represents a random transposition of the form $(k, k \oplus 2^i)^b$, where parallel lines indicate $b = 0$ and a crossover indicates $b = 1$

To overcome the limitations of having so few permutations, our strategy is as follows: the goal at the end of round 1, is to collect integers with leading 1 into the lowest part of the bottom half (those positions ≤ 3), and to collect integers with leading 0 into the lowest part of the top half (those positions ≥ 4). Then the powers of the transpositions in the final round (round 2) are determined by the sorting requirement. We claim that this strategy will work for all n .

4 The Construction Details

In the next two subsections we present the detailed construction of the cipher X of Lemma 3. In the following two subsections we prove Lemma 3 and sketch the proof of Lemma 2, respectively.

4.1 Algebraic Details

We may encrypt by inverting the sorting procedure described in the previous section. Formally, for any j , define $R_i^{(j)}$ to be the product of independent and uniformly random powers of the 2^{n-1} distinct transpositions of the form $(k, k \oplus 2^i)$, with $0 \leq k \leq 2^n - 1$. Then let

$$P_i = R_0^{(i)} R_1^{(i)} \cdots R_{i-1}^{(i)},$$

and as before $X = P_n \cdots P_2 P_1$. Each random involution $R_i^{(j)}$ corresponds to a “round” as shown in Fig. 1 below. Note that while there is repetition (e.g. $R_i^{(j_1)}$ and $R_i^{(j_2)}$ are i.i.d. random variables), X is *not* a traditional iterated cryptosystem because the specific sequence of rounds is carefully chosen.

4.2 Algorithm Details

It is clear that we may recursively affect the actions of X and X^{-1} on any block, if we can carry out the rounds $R_i^{(j)}$ in the correct order and in such a way that the powers of all relevant transpositions are independent and equiprobable. Moreover, if we encounter the the same butterfly in two different executions, we must be able to reproduce the same random power of the corresponding transposition. This is easily accomplished if we consider the random bits to be indexed by $\mathcal{M} \times \mathbb{Z}$. The resulting function $f : \mathcal{M} \times \mathbb{Z} \rightarrow \{0, 1\}$ is easily transformed into a random oracle function $F : \{0, 1\}^* \rightarrow \{0, 1\}$ appropriate for Definition 1. We employ the convention that the power of any transposition $(k, k \oplus 2^i)$ is $f(m, r)$, where $m = \min\{k, k \oplus 2^i\}$ and r is the round. In other words, f is applied to the lower left hand corner of every butterfly in Fig. 1.

The next algorithm implements encryption. To encrypt a single plaintext block, the computational complexity will be $n(n+1)/2$ or $O(\frac{1}{2}n^2)$. For a block size of $n = 128$, this yields about 8,256 operations.

Algorithm 1. *Defines recursive encryption functions Enc and invSort. The action of $X = P_n \cdots P_2 P_1$ on $p \in \mathcal{M}$ is affected by $(q, r) = \text{Enc}(p, n, 1)$, such that $q = Xp$. The action of P_i on $p \in \mathcal{M}$ is affected by $(q, r) = \text{invSort}(p, i - 1, *)$, such that $q = P_i p$.*

| | |
|--|--|
| <pre> function Enc(p, i, r): if $i > 1$ then (q, r) = Enc($p, i - 1, r$). endif return invSort($q, i - 1, r$). </pre> | <pre> function invSort(p, j, r): $q = p \oplus 2^j$. if $p < q$ then $b = f(p, r)$. else $b = f(q, r)$. endif if $b = 0$ then $q = p$. endif if $j > 0$ then return invSort($q, j - 1, r + 1$). else return ($q, r + 1$). endif </pre> |
|--|--|

The decryption algorithm is easily obtained by performing the the transpositions in the reverse order. The necessary modifications are immediate, and we shall call the “reverse” of inverse-sorting fwdSort.

Algorithm 2. *Defines recursive decryption functions Dec and fwdSort. The action of $X^{-1} = P_1^{-1} P_2^{-1} \cdots P_n^{-1}$ on $p \in \mathcal{M}$ is affected by $(q, r) = \text{Dec}(p, n, \frac{1}{2}n(n+1))$, such that $q = X^{-1}p$. The action of P_i^{-1} on $p \in \mathcal{M}$ is affected by $(q, r) = \text{fwdSort}(p, i - 1, *)$, such that $q = P_i^{-1}p$.*

| | |
|---|---|
| <pre> function Dec(p, i, r): (q, r) = fwdSort($q, i - 1, r$). if $i > 1$ then return Dec($q, i - 1, r$). else return (q, r). endif </pre> | <pre> function fwdSort(p, j, r): if $j > 0$ then (p, r) = fwdSort($p, j - 1, r$). endif $q = p \oplus 2^j$. if $p < q$ then $b = f(p, r)$. else $b = f(q, r)$. endif if $b = 0$ then return ($p, r - 1$). else return ($q, r - 1$). endif </pre> |
|---|---|

Remark 1. Notice how the round information is explicitly carried by input–output argument r through the entire recursion processed. During the execution of **Enc**, it is incremented, while during the execution of **Dec** it is decremented. This is necessary because encryption and decryption must agree on the random bits $f(p, r)$ which determine the appropriate powers of the various transpositions involved. \square

4.3 The Proof of Lemma 3

To prove Lemma 3 we must first develop some terminology and prove some preliminary results. Recall that the integers in \mathcal{M} will have a dual role as n -bit strings. When treating prefixes and other substrings it is useful to have a *padding function* $\pi_i : \mathbb{Z} \rightarrow \{0, 1\}^i$ taking j to the binary representation of $j \bmod 2^i$ padded up to i bits. Also define a *prefix truncation function* $\tau_i : \{0, 1\}^* \rightarrow \{0, 1\}^i$ taking binary word w to its first i bits (the most significant i bits).

It is natural for us to recursively partition \mathcal{M} into disjoint subsets which share the same prefix. For example, let $S_0 = \{i \in \mathcal{M} \mid \tau_1(i) = 0\}$ and $S_1 = \{i \in \mathcal{M} \mid \tau_1(i) = 1\}$, so that \mathcal{M} is the disjoint union $S_0 \cup S_1$. More generally, let $S_{\pi_i(j)} = \{k \in \mathcal{M} \mid \tau_i(k) = \pi_i(j)\}$ with $1 \leq j \leq 2^i - 1$, and again we partition \mathcal{M} into disjoint subsets

$$\mathcal{M} = \bigcup_{j=0}^{2^i-1} S_{\pi_i(j)}.$$

The prefix stabilizers are naturally expressed in terms of these subsets, for example clearly $K_{n-1} = \text{Stab}_{K_n}\{S_0\} \cap \text{Stab}_{K_n}\{S_1\}$, and more generally

$$K_{n-i} = \bigcap_{j=0}^{2^i-1} \text{Stab}_{K_n}\{S_{\pi_i(j)}\}.$$

The following proposition characterizes the left cosets of $K_{n-1} \leq K_n$.

Proposition 1. *A left coset of K_{n-1} in K_n is completely determined by the image of S_0 under the action of any left coset representative.*

Proof. First of all $K_{n-1} = \text{Stab}_{K_n}\{S_0\} \cap \text{Stab}_{K_n}\{S_1\} = \text{Stab}_{K_n}\{S_0\}$, because anything which fixes S_0 must also fix S_1 . Now $K_n = \mathfrak{S}_{\mathcal{M}}$ acts transitively on the set $\mathcal{M}^{\{2^{n-1}\}}$ of all subsets of \mathcal{M} of half its size. By standard group action arguments [19,7], the left cosets $\{gK_{n-1}\}$ are in one-to-one correspondence with the images $\{gS_0\}$, in a well-defined way. \square

We shall derive presently a similar characterization of the left cosets of K_{n-i-1} in K_{n-i} . First let's agree that whenever $A \subset B$ we will consider \mathfrak{S}_A to be a subgroup of \mathfrak{S}_B . Recall [19] that if a group G factors into product $G = HK$ of normal subgroups H and K , with $H \cap K = \{1\}$, then G is a *direct product* of H and K (it is literally isomorphic to the Cartesian product with the obvious group law). Clearly whenever B is a disjoint union of A_1 and A_2 , \mathfrak{S}_B contains the direct product $\mathfrak{S}_{A_1}\mathfrak{S}_{A_2}$. Visibly, $K_{n-1} = \mathfrak{S}_{S_0}\mathfrak{S}_{S_1}$, and if we write $\mathfrak{S}_{\pi_i(j)} = \mathfrak{S}_{S_{\pi_i(j)}}$, we also have that K_{n-i} is the direct product

$$K_{n-i} = \prod_{j=0}^{2^i-1} \mathfrak{S}_{\pi_i(j)}.$$

Proposition 2. *A left coset of K_{n-i-1} in K_{n-i} is completely determined by the images of $S_{\pi_i(j)0}$, $0 \leq j \leq 2^i-1$, under the action of any left coset representative.*

Proof. Because K_{n-i} is the direct product given above, a left coset gK_{n-i-1} factors into a product of left cosets

$$\prod_{j=0}^{2^i-1} g_j \left(\text{Stab}_{\mathfrak{S}_{\pi_i(j)}}\{S_{\pi_i(j)0}\} \cap \text{Stab}_{\mathfrak{S}_{\pi_i(j)}}\{S_{\pi_i(j)1}\} \right).$$

However, we again have

$$\text{Stab}_{\mathfrak{S}_{\pi_i(j)}}\{S_{\pi_i(j)0}\} \cap \text{Stab}_{\mathfrak{S}_{\pi_i(j)}}\{S_{\pi_i(j)1}\} = \text{Stab}_{\mathfrak{S}_{\pi_i(j)}}\{S_{\pi_i(j)0}\}.$$

Finally, 2^i invocations of Prop. 1 obtains the desired result. \square

With this machinery in place, we may now prove Lemma 3.

Proof (of Lemma 3). Recall that in order to facilitate the induction argument of Sect. 3.2, thereby establishing that $\text{supp}(X) = \mathfrak{S}_{\mathcal{M}}$, we must show that (for each i) $\text{supp}(P_{n-i})$ contains a representative of each left coset of K_{n-i-1} in K_{n-i} .

What we'll actually show, by an inner induction argument, is that for every subset $S \subset \mathcal{M}$ contiguous on each $S_{\pi_i(j)}$ ($0 \leq j \leq 2^i-1$) and every possible image T of S under the action of K_{n-i} (i.e., every T of the form gS for some $g \in K_{n-i}$), $\text{supp}(P_{n-i})$ contains a permutation g taking $S \mapsto T$. Since each $S_{\pi_i(j)0}$ is trivially a contiguous subset of $S_{\pi_i(j)}$, we have the desired result by

Prop. 2. Note also that if we can take an arbitrary contiguous set to an arbitrary image, then we can also take an arbitrary complement of a contiguous set to an arbitrary image.

Induction Base: Clearly K_1 is isomorphic to the direct product of 2^{n-1} symmetric groups on 2 elements (cyclic groups of order 2), and thus has size $|K_1| = 2^{2^{n-1}}$. Since $|\text{supp}(P_1)| = 2^{2^{n-1}}$ also, the induction hypothesis holds trivially.

Induction Step: Without loss of generality, we consider the case $i = 0$. By hypothesis, $\text{supp}(P_{n-1})$ contains an element of K_{n-1} taking any contiguous subset of S_0 to a desired image ($\subset S_0$, and of the same size), while simultaneously taking any contiguous subset of S_1 to a desired image (again $\subset S_1$, and of the same size). Choose arbitrary sets $U \subset S_0, V \subset S_1$, let $T = U \cup V$, and choose any contiguous set $S \subset \mathcal{M}$ of size $|T|$. Again without loss of generality, we may assume that $|S \cap S_0| \geq |U|$ (because otherwise $|S \cap S_1| \geq |V|$ and a completely symmetric argument applies). We must show that $\text{supp}(P_n) = \text{supp}(P_{n-1})\text{supp}(R_{n-1}^{(n)})$ contains a g such that $gS = T$. Write $g = hka$, with $h \in \mathfrak{S}_{S_0}, k \in \mathfrak{S}_{S_1}$, and where a is some product of transpositions of the form $(j, j \oplus 2^{n-1})$. Evidently the real job of a is to send elements of $S \cap S_0$ in excess of $|U|$ across the most significant bit boundary into S_1 , because $h, k \in \text{Stab}_{K_n}\{S_0\}$ cannot do this later on. The transpositions in $\text{supp}(R_{n-1}^{(n)})$, which flip the most significant bit, are perfect for this task. Let a be the product of the transpositions $(j, j \oplus 2^{n-1})$, with $j \in J$, where J consists of the highest $|S \cap S_0| - |U|$ elements of $S \cap S_0$. We claim that $(aS) \cap S_0$ is a contiguous subset of S_0 , and that $(aS) \cap S_1$ is either a contiguous subset or the complement of a contiguous subset of S_1 . Assuming that is true, then by the induction hypothesis, we may choose h taking $(aS) \cap S_0 \mapsto U$ and k taking $(aS) \cap S_1 \mapsto V$, so that $gS = hkaS = T$.

Two cases naturally arise. (*Case 1:*) If S doesn't intersect with S_1 then a takes J contiguously to some image in the middle of S_1 , and a leaves $S - J$ contiguously in the middle of S_0 . (*Case 2:*) On the other hand, if S intersects non-trivially with S_1 , then because S is contiguous, J is precisely the highest $|S \cap S_0| - |U|$ elements of S_0 itself, and furthermore $S \cap S_1$ consists of the lowest $|S \cap S_1|$ elements of S_1 which are left fixed by a . Therefore $(aS) \cap S_1$ consists of the complement of a contiguous set (those elements between $S \cap S_1$ and aJ). But again a leaves $(S \cap S_0) - J$ contiguously in the middle of S_0 . This completes the induction step for $i = 0$.

Applying this same argument within the appropriate direct product subgroups when $i > 0$ yields the inner induction step and thus completes the proof. \square

Remark 2. The previous proof seems harrowing with 2 cases nested inside 2 w.l.o.g.'s nested inside of 2 layers of induction. But, it is in essence just a rigorous form of the more intuitive sorting example given in the previous section (which may have seemed simpler at first glance). \square

4.4 Sketch of the Proof of Lemma 2

In this section, we shall sketch the proof of Lemma 2 with the help of some preliminary results.

The following inequality from [17] is essentially the translation into guesswork terminology of a nice result attributed to Day [5] about majorization for sums of real vectors (see [13] and [17]). Given any m vectors $x^{(1)}, \dots, x^{(m)} \in \mathbb{R}_+^n$, m doubly stochastic $n \times n$ matrices D_1, \dots, D_m , and m positive real numbers $\omega_1, \dots, \omega_m$, we have

$$W\left(\sum_{i=1}^m \omega_i D_i x^{(i)}\right) \geq \sum_{i=1}^m \omega_i W(x^{(i)}). \quad (7)$$

Our technique for quantifying and comparing various performance values of the optimal chosen plaintext attack typically starts with a fixed ℓ and fixed $p \in \mathcal{M}^{(\ell)}$. We then proceed to study how the cipher's structure affects the expression of (3). A simple but useful observation is that the conditional guesswork $W(Y|c, p)$ is completely determined by the distribution of Y on some coset of the stabilizer $H = \text{Stab}_G(p)$. Let $k = [G : H]$ and fix a set $\{g_i\}_{i=1}^k$ of left coset representatives of H in G . It is useful to treat the distribution $y(g) = \mathbb{P}[Y = g]$ as giant vector in $\mathbb{R}G$ (the real vector space spanned by G) which decomposes into a direct sum of left coset component vectors $\{y^{(i)}\}_{i=1}^k$, as described below.

A mathematically succinct way to handle this decomposition of y , especially when we need to study the effect of multiplication by X , is to exploit the fact that the vector space $\mathbb{R}G$ is also the real group algebra generated by G and is isomorphic as a left $\mathbb{R}G$ -module to the *induced representation from H to G by $\mathbb{R}H$* given by the tensor product of modules⁴

$$\mathbb{R}G \cong \mathbb{R}G \otimes_{\mathbb{R}H} \mathbb{R}H = \bigoplus_{i=1}^k g_i \otimes \mathbb{R}H, \quad (8)$$

⁴ Our treatment of induced representations follows Jacobson's [12] and is aimed at succinctness. Briefly, given a ring B , a right B -module U , and a left B -module V , one forms the *tensor product of modules* $T = U \otimes_B V$ in a way which is completely analogous to the case of vector spaces, except that now we have $ub \otimes v = u \otimes bv$, $b \in B$. In general, T is only a \mathbb{Z} -module, but when U is a A - B -bimodule for some ring A (i.e. U is a left A -module as well as a right B -module), T becomes a left A -module with left multiplication defined by $a(u \otimes v) = au \otimes v$. In this way, for any $\mathbb{R}H$ -module V , $\mathbb{R}G \otimes_{\mathbb{R}H} V$ becomes a $\mathbb{R}G$ -module called the *representation induced from H to G by V* .

The reader who is unfamiliar with the module approach to representations is encouraged to begin with a more constructive definition of the induced representation (such as in [10]) and work out the relatively inelegant details for left multiplication by a cipher X .

where the isomorphism takes $g_i h \mapsto g_i \otimes h$, and thus takes

$$\sum_{g \in G} y(g)g = \sum_{i=1}^k \sum_{h \in H} y(g_i h)g_i h \mapsto \hat{y} = \sum_{i=1}^k g_i \otimes y^{(i)} = \sum_{i=1}^k g_i \otimes \left(\sum_{h \in H} y(g_i h)h \right). \quad (9)$$

Note that $\mathbf{P}[Y = g_i h] = y(g_i h) = y_h^{(i)}$, and so the vectors $\{y^{(i)}\}$ really describe the distribution of Y on the cosets of H , even though each one is an element of $\mathbb{R}H$. Now, it follows directly from the definition that

$$W(Y|C^\ell, p) = \sum_{i=1}^k W(y^{(i)}).$$

Now recall that the product $Z = XY$ of two independent G -ciphers X and Y has distribution $\mathbf{P}[XY = g] = x * y(g)$, which from (1) has the form of a matrix multiplication. Indeed using the direct sum decomposition of the induced representation given in (8), we shall derive the block structure of this matrix. Using this structure we shall compare the distribution within the appropriate cosets of H for XY vs. Y . The key is to represent Y by \hat{y} as in (9), but to leave X as a convex sum of the permutations in G weighted by $x(g) = \mathbf{P}[X = g]$. We aim to derive the form of Z represented by $\hat{z} \in \mathbb{R}G \otimes_{\mathbb{R}H} \mathbb{R}H$, again as in (9).

Now any $g \in G$ acts by left multiplication on any $g_j \otimes v \in \mathbb{R}G \otimes_{\mathbb{R}H} \mathbb{R}H$ according to $g(g_j \otimes v) = g_i \otimes hv$, where $gg_j H = g_i H$, so that $h \in H$ is uniquely determined by $gg_j = g_i h$. Thus we have that

$$\begin{aligned} \hat{z} &= \sum_{i=1}^k g_i \otimes z^{(i)} = \left(\sum_{g \in G} x(g)g \right) \left(\sum_{j=1}^k g_j \otimes y^{(j)} \right) \\ &= \sum_{j=1}^k \left(\sum_{g \in G} x(g)g \right) (g_j \otimes y^{(j)}). \end{aligned}$$

For any particular i we may collect together contributions to direct summand $g_i \otimes \mathbb{R}H$,

$$\begin{aligned} g_i \otimes z^{(i)} &= \sum_{j=1}^k \sum_{g \in G, g_j H = g_i H} x(g)g (g_j \otimes y^{(j)}) \\ &= \sum_{j=1}^k \sum_{g \in \Gamma_{ij}} x(g)(g_i \otimes h_{ij}(g)y^{(j)}) \\ &= g_i \otimes \left(\sum_{j=1}^k \left(\sum_{g \in \Gamma_{ij}} x(g)h_{ij}(g) \right) y^{(j)} \right), \end{aligned}$$

where $\Gamma_{ij} = \{g \in G \mid g g_j H = g_i H\}$ and $h_{ij}(g) = g_i^{-1} g g_j$. Thus,

$$z^{(i)} = \sum_{j=1}^k \omega_{ij} D_{ij} y^{(j)}, \quad (10)$$

where

$$D_{ij} = \sum_{g \in \Gamma_{ij}} \frac{x(g)}{x(\Gamma_{ij})} h_{ij}(g), \quad (11)$$

and where $\omega_{ij} = x(\Gamma_{ij})$. Notice that the sum in (11) is a convex sum of permutations in H , hence each D_{ij} takes the form of a doubly stochastic matrix under a suitable ordering of H (the basis vectors of $\mathbb{R}H$). Furthermore, it can easily be shown that the values of ω_{ij} are the elements of a doubly stochastic matrix [17]. Also note that $\Gamma_{ii} = H^{g_i} \leq G$ for each i . The true core of Lemma 2 is the following proposition.

Proposition 3. *For a permutation group $G \leq \mathfrak{S}_{\mathcal{M}}$, let X and Y be independent G -ciphers such that $\text{supp}(X) = G$. For any $p \in \mathcal{M}^{(\ell)}$ such that Y is non-uniform on at least one left coset of $\text{Stab}_G(p)$, we have $W(XY|C^\ell, p) > W(Y|C^\ell, p)$.*

Proof. Let p satisfy the assumption of the proposition, and let \hat{z} represent the distribution of the product $Z = XY$ as above. Let $y^{(j)}$ be non-uniform and consider $D_{jj}y^{(j)}$. That is to say, let us focus on this one submatrix block on the diagonal of the larger doubly stochastic matrix representing the convolution $z = x * y$.

Since $\Gamma_{jj} = H^{g_j}$, we may rewrite D_{jj} as

$$D_{jj} = \sum_{g \in \Gamma_{jj}} \frac{x(g)}{x(\Gamma_{jj})} g^{g_j^{-1}} = \sum_{h \in H} \frac{x(h^{g_j})}{x(H^{g_j})} h.$$

Thus we see that by scaling appropriately, $D_{jj}y^{(j)}$ has the form of a product of two independent H -ciphers $\tilde{X}\tilde{Y}$, with $\mathbf{P}[\tilde{X} = h] = \mathbf{P}[X = h^{g_j}] / x(H^{g_j})$, and $\mathbf{P}[\tilde{Y} = h] = \mathbf{P}[Y = g_j h] / y(g_j H)$. But since $\text{supp}(X) = G$ and conjugation by g_j yields an isomorphism of $H \longleftrightarrow \Gamma_{jj}$, $\text{supp}(\tilde{X})$ is not confined to any proper coset of H , and we may invoke Lemma 1 to obtain $W(\tilde{X}\tilde{Y}) > W(\tilde{Y})$ or more importantly for our purposes, $W(D_{jj}y^{(j)}) > W(y^{(j)})$.

Note that we may bound any $W(z^{(i)})$ by the inequality of (7) as

$$W(z^{(i)}) = W\left(\sum_{m=1}^k \omega_{im} D_{im} y^{(m)}\right) \geq \sum_{m=1}^k \omega_{im} W(y^{(m)}),$$

but by using $W(D_{jj}y^{(j)}) > W(y^{(j)})$ and (7) again, we may strictly bound $W(z^{(j)})$ as follows

$$\begin{aligned} W(z^{(j)}) &= W\left(\sum_{m=1}^k \omega_{jm} D_{jm} y^{(m)}\right) \\ &\geq \sum_{m=1}^k \omega_{jm} W(D_{jm} y^{(m)}) \\ &> \sum_{m=1}^k \omega_{jm} W(y^{(m)}). \end{aligned}$$

(Note that in both cases we have used, for doubly stochastic D , the inequality $W(Dv) \geq W(v)$ which follows from simple majorization arguments [17]). Combining these bounds on $W(z^{(i)})$ we obtain a strict bound on $W(Z|C^\ell, p)$ as follows

$$\begin{aligned}
 W(Z|C^\ell, p) &= \sum_{i=1}^k W(z^{(i)}) > \sum_{i=1}^k \sum_{m=1}^k \omega_{im} W(y^{(m)}) \\
 &= \sum_{m=1}^k W(y^{(m)}) \sum_{i=1}^k \omega_{im} \\
 &= \sum_{m=1}^k W(y^{(m)}) = W(Y|C^\ell, p),
 \end{aligned}$$

which was to be proved. \square

Remark 3. Evidently, in the previous proposition, we could weaken the condition $\text{supp}(X) = G$ to: For every $p \in \mathcal{M}^{(\ell)}$, $\text{supp}(X) \cap \text{Stab}_G(p)$ is not confined to a proper coset of $\text{Stab}_G(p)$. However, for our purposes in this paper, it was not necessary to use the weaker condition. \square

The next proposition provides an important interpretation of the situation when a cipher is uniform on every coset of an ℓ -message stabilizer.

Proposition 4. *Let Y be a G -cipher, for a permutation group $G \leq \mathfrak{S}_{\mathcal{M}}$. For any $p \in \mathcal{M}^{(\ell)}$, write $H = \text{Stab}_G(p)$ and we have*

$$W(Y|C^\ell, p) \leq \frac{1 + |H|}{2},$$

with equality holding iff Y is uniform on each coset of H .

Proof. For $c \in \mathcal{M}^{(\ell)}$ with $\omega(c|p) \neq 0$,

$$1 \leq W(Y|c, p) \leq \frac{1 + |H|}{2},$$

because $W(Y|c, p)$ is the guesswork on a coset of size $|H|$. Furthermore, equality in the upper bound is achieved iff Y has constant probability on that particular coset [17]. Now since $\sum_{c \in \mathcal{M}^{(\ell)}} \omega(c|p) = 1$, the sum from (3)

$$W(Y|C^\ell, p) = \sum_{c \in \mathcal{M}^{(\ell)}} W(Y|c, p) \omega(c|p)$$

is convex and therefore achieves its maximum of $\frac{1}{2}(1 + |H|)$ iff Y is constant on each coset of H (Y will of course have the constant probability 0 on those cosets corresponding to $\omega(c|p) = 0$). \square

By tying together the previous two propositions, we may finally prove Lemma 2.

Proof (of Lemma 2). Again, let us write $Z = XY$. Suppose there is a $p \in \mathcal{M}^{(\ell)}$ such that $\theta_\ell(Z) = W(Z|C^\ell, p)$ and Y is non-uniform on at least one coset of $\text{Stab}_G(p)$. Then we may invoke Prop. 3 to obtain

$$\theta_\ell(Z) = W(Z|C^\ell, p) > W(Y|C^\ell, p) \geq \theta_\ell(Y).$$

On the other hand, suppose that for every $p \in \mathcal{M}^{(\ell)}$ satisfying $\theta_\ell(Z) = W(Z|C^\ell, p)$, Y is uniform on each coset of $\text{Stab}_G(p)$. Let $H = \text{Stab}_G(p)$ for any such p . By (10), Z is uniform on each coset of H as well, and by Prop. 4,

$$\theta_\ell(Z) = W(Z|C^\ell, p) = \frac{1 + |H|}{2}$$

Now choose any \hat{p} with $|\text{Stab}_G(\hat{p})| = M_G(\ell)$ and hence

$$\theta_\ell(Z) = \frac{1 + |H|}{2} \leq W(Z|C^\ell, \hat{p}) \leq \frac{1 + M_G(\ell)}{2},$$

forcing $|H| = M_G(\ell)$, and thus $\theta_\ell(Z) = \theta_\ell(U_G)$. Then, either $\theta_\ell(Y) \neq \theta_\ell(U_G)$, in which case $\theta_\ell(Z) > \theta_\ell(Y)$, or $\theta_\ell(Y) = \theta_\ell(U_G)$.

To summarize what we have proved thus far, $\theta_\ell(Z) \geq \theta_\ell(Y)$ and if equality holds then $\theta_\ell(Y) = \theta_\ell(U_G)$. However conversely, if $\theta_\ell(Y) = \theta_\ell(U_G)$, then $\theta_\ell(U_G) \geq \theta_\ell(Z) \geq \theta_\ell(Y) = \theta_\ell(U_G)$, forcing equality $\theta_\ell(Z) = \theta_\ell(Y)$, which completes the proof. \square

5 Conclusion

The issue of security amplification by product composition remains a complex one. In this paper, we have added to the number of situations where a definite answer can be given. Specifically, Theorem 1 asserts that there exists efficient cipher X such that the security of XY is strictly greater than Y unless Y is perfect. There is room for further improvement in this result. For example, a more efficient cipher might be constructed which makes use of a weakened form of Lemma 2 as discussed in Remark 3. Additionally, our implementation might be optimized for bulk encryption.

The cipher we construct to prove Theorem 1 is costly in some ways but has other desirable properties. Unlike a one-time pad, if the key were replaced by a pseudo-random source, a known plaintext-ciphertext block would *not* trivially betray the key used for that block. This property could be useful in constructing provably secure practical encryption systems. Also observe that our construction is *not* an iterated cryptosystem but rather a product of independent rounds with a *carefully chosen order*. The techniques employed here might be a useful new paradigm for practical cryptosystems with key schedules instead of a truly random source of key material.

Finally we note that due to Lemma 2 and the nature of our existence question we have been content to focus on strict inequalities and strict amplification of support alone. While an infinitesimally small increase from $\theta_\ell(Y)$ to $\theta_\ell(XY)$ is possible, techniques beyond the scope of this paper have been developed to establish much stronger claims of amplification. Ongoing research suggests that the cipher X of Lemma 3 has stronger security properties than required by Theorem 1.

Acknowledgments

I would like to thank Serge Vaudenay for his many insightful comments, and in particular for suggesting the formal computational model of Sect. 2.

References

1. W. Aiello, M. Bellare, G. Di Crescenzo, and R. Venkatesan. Security amplification by composition: The case of doubly-iterated, ideal ciphers. In H. Krawczyk, editor, *Advances in Cryptology - CRYPTO '98*, Berlin, 1998. Springer-Verlag.
2. David J. Aldous and Persi Diaconis. Shuffling cards and stopping times. *Amer. Math. Monthly*, 93:333–348, 1986.
3. Christian Cachin. *Entropy Measures and Unconditional Security in Cryptography*. PhD thesis, ETH Zürich, 1997.
4. Keith W. Campbell and Michael J. Wiener. DES is not a group. In Ernest F. Brickell, editor, *Advances in Cryptology - CRYPTO '92*, pages 512–517, Berlin, 1992. Springer-Verlag.
5. P. W. Day. Rearrangement inequalities. *Canad. J. Math.*, 24:930–943, 1972.
6. Persi Diaconis. *Group Representations in Probability and Statistics*. Institute of Mathematical Statistics, Hayward, CA, 1988.
7. John D. Dixon and Brian Mortimer. *Permutation Groups*. Springer-Verlag, New York, 1996.
8. S. Even and O. Goldreich. On the power of cascade ciphers. *ACM Transactions on Computer Systems*, 3(2), 1985.
9. Michael R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman and Company, New York, 2nd edition, 1979.
10. Roe Goodman and Nolan R. Wallach. *Representations and Invariants of the Classical Groups*, volume 68 of *Encyclopedia of Mathematics and its Applications*. Cambridge University Press, Cambridge, 1998.
11. G. R. Grimmett and D. R. Stirzaker. *Probability and Random Processes*. Oxford University Press, Oxford, 2nd edition, 1992.
12. Nathan Jacobson. *Basic Algebra II*. W. H. Freeman and Company, New York, 2nd edition, 1980.
13. Albert W. Marshall and Ingram Olkin. *Inequalities: Theory of Majorization and Its Applications*. Academic Press, San Diego, 1979.
14. Ueli M. Maurer and James L. Massey. Cascade ciphers: The importance of being first. *Journal of Cryptology*, 6:55–61, 1993.
15. Alfred J. Menezes, Paul C. van Oorschot, and Scott A. Vanstone. *Handbook of Applied Cryptography*. CRC Press, Boca Raton, 1997.

16. Moni Naor and Omer Reingold. On the construction of pseudorandom permutations: Luby-Rackoff revisited. *Journal of Cryptology*, 12:29–66, 1999.
17. John O. Pliam. *Ciphers and their Products: Group Theory in Private Key Cryptography*. PhD thesis, University of Minnesota, July 1999. URL: <http://www.ima.umn.edu/~pliam/doc>.
18. John O. Pliam. Guesswork and variation distance as measures of cipher security. In *Selected Areas in Cryptography - SAC'99*, LNCS 1758, pages 62–77, Berlin, 2000. Springer-Verlag.
19. Joseph J. Rotman. *An Introduction to the Theory of Groups*. Wm. C. Brown, Dubuque, IA, 3rd edition, 1988.
20. Claude E. Shannon. Communication theory of secrecy systems. *Bell System Tech. Jour.*, 28:656–715, 1949.

Decorrelation over Infinite Domains: The Encrypted CBC-MAC Case

Serge Vaudenay

Swiss Federal Institute of Technology (EPFL)
`Serge.Vaudenay@epfl.ch`

Abstract. Decorrelation theory has recently been proposed in order to address the security of block ciphers and other cryptographic primitives over a finite domain. We show here how to extend it to infinite domains, which can be used in the Message Authentication Code (MAC) case.

In 1994, Bellare, Kilian and Rogaway proved that CBC-MAC is secure when the input length is fixed. This has been extended by Petrank and Rackoff in 1997 with a variable length.

In this paper, we prove a result similar to Petrank and Rackoff's one by using decorrelation theory. This leads to a slightly improved result and a more compact proof.

This result is meant to be a general proving technique for security, which can be compared to the approach which was announced by Maurer at CRYPTO'99.

Decorrelation theory has recently been introduced. (See references [17] to [22].) Its first aim was to address provable security in the area of block ciphers in order to prove their security against differential [7] and linear cryptanalysis [10]. As a matter of fact, these techniques have also been used in order to prove Luby-Rackoff -like pseudorandomness results [9] in a way similar to Patarin's "coefficient H method" [14,15]. All previous cases however address random functions over a finite domain, which is not appropriate for MACs.

The CBC-MAC construction is well known in order to make Message Authentication Codes from a block cipher in Cipher Block Chaining mode. Namely, if C is a permutation defined on a block space $\{0,1\}^m$, for a message $x = (m_1, \dots, m_\ell) \in (\{0,1\}^m)^\ell$ we define

$$\text{MAC}(x) = C(C(\dots C(m_1) + m_2 \dots) + m_\ell).$$

The addition is traditionally the XOR operation but can be replaced by any group (or even quasigroup) law. In 1994, Bellare, Kilian and Rogaway proved that if C is a uniformly distributed random permutation, then for any integer ℓ and any distinguisher between MAC and a truly random function which is limited to d queries, the advantage is less than $3d^2\ell^22^{-m}$ [6]. This shows that no adaptive attack can forge a new valid $(x, \text{MAC}(x))$ pair with a relevant probability unless the total number of known blocks $d\ell$ is within the order of $2^{\frac{m}{2}}$. This however holds when all messages have the fixed length ℓ . If the attacker is

allowed to use messages with different length, it is easy to notice that for any message x and any block a the MAC of x concatenated with $a - \text{MAC}(x)$ is

$$\text{MAC}(x, a - \text{MAC}(x)) = C(a)$$

which does not depend on x and allows to forge a new authenticated message by replacement of x .

In 1997, Petrank and Rackoff addressed the case of DMAC defined by

$$\text{MAC}(x) = C_2(C_1(C_1(\dots C_1(m_1) + m_2 \dots) + m_\ell))$$

(see [16]). This type of construction does not mean any originality since it is already suggested by several standards [2,3,4]. Its security was however formally proved in [16] for the first time.

If we replace C_2 by $C_2 \circ C_1^{-1}$ we can obviously remove the last C_1 application. We can thus consider the MAC defined by

$$\text{MAC}(x) = C_2(C_1(\dots C_1(m_1) + m_2 \dots) + m_\ell)$$

which we call the “encrypted CBC-MAC” in the sequel. In this paper we give a security proof which is different from [16] and with a slightly improved reduction. Our proof also happens to be more compact (it is less than 2-page long), thanks to use of the decorrelation theory tools. Our approach is also more general and can be applied to other schemes. In this way it can be compared to the information theoretic general approach which was announced by Maurer at CRYPTO’99 [12].

1 Prerequisite

1.1 Definitions and Notations

First of all, for any random function F from a set \mathcal{M}_1 to a set \mathcal{M}_2 and any integer d we associate the “ d -wise distribution matrix” which is denoted $[F]^d$, defined in the matrix set $\mathbf{R}^{\mathcal{M}_1^d \times \mathcal{M}_2^d}$ by

$$[F]_{(x_1, \dots, x_d), (y_1, \dots, y_d)}^d = \Pr[F(x_1) = y_1, \dots, F(x_d) = y_d].$$

Given a metric structure D in $\mathbf{R}^{\mathcal{M}_1^d \times \mathcal{M}_2^d}$ we can define the distance between the matrices associated to two random functions F and G . This is the “ d -wise decorrelation distance”. If G is a random function uniformly distributed in the set of all functions from \mathcal{M}_1 to \mathcal{M}_2 (we let F^* denote such a function), this distance is called the “ d -wise decorrelation bias of function F ” and denoted $\text{DecF}_D^d(F)$. When F is a permutation (which will usually be denoted C as for “Cipher”) and G is a uniformly distributed permutation (denoted C^*) it is called the “ d -wise decorrelation bias of permutation F ” and denoted $\text{DecP}_D^d(F)$. In previous results we used the metric structures defined by the norms denoted $\|\cdot\|_2$ (see

[18]), $|||\cdot|||_\infty$, $||\cdot||_a$, $||\cdot||_s$ (see [21]). These four norms are matrix norms, which means that they are norms on $\mathbf{R}^{\mathcal{M}_1^d \times \mathcal{M}_2^d}$ with the property that

$$||A \times B|| \leq ||A|| \cdot ||B||.$$

This property leads to non-trivial inequalities which can shorten many treatments on the security of conventional cryptography.

Given two random functions F and G from \mathcal{M}_1 to \mathcal{M}_2 we call “distinguisher between F and G ” any oracle Turing machine \mathcal{A}^O which can send \mathcal{M}_1 -element queries to the oracle O and receive \mathcal{M}_2 -element responses, and which finally outputs 0 or 1. In particular the Turing machine can be probabilistic. In the following, the number of queries to the oracle will be limited to d . The distributions on F and G induces a distribution on \mathcal{A}^F and \mathcal{A}^G , thus we can compute the probability that these probabilistic Turing machines output 1. The advantage for distinguishing F from G is

$$\text{Adv}_{\mathcal{A}}(F, G) = \Pr[\mathcal{A}^F \rightarrow 1] - \Pr[\mathcal{A}^G \rightarrow 1].$$

For any class of distinguishers Cl we will denote

$$\text{Adv}_{\text{Cl}}(F, G) = \max_{\mathcal{A} \in \text{Cl}} \text{Adv}_{\mathcal{A}}(F, G).$$

We notice that if \mathcal{A} is a distinguisher, we can always define a complementary distinguisher $\bar{\mathcal{A}} = 1 - \mathcal{A}$ which gives the opposite output. There is no need for investigating the minimum advantage when the class is closed under the complement (which is the case of the above class) since

$$\text{Adv}_{\bar{\mathcal{A}}}(F, G) = -\text{Adv}_{\mathcal{A}}(F, G).$$

We consider the class Cl_a^d of all (adaptive) distinguishers limited to d queries.

1.2 Properties

The d -wise distribution matrices have the property that if F and G are independent random functions, F from \mathcal{M}_2 to \mathcal{M}_3 and G from \mathcal{M}_1 to \mathcal{M}_2 , then

$$[F \circ G]^d = [G]^d \times [F]^d.$$

Thus, if we are using a matrix norm $||\cdot||$, we obtain

$$\text{DecF}_{||\cdot||}^d(F \circ G) \leq \text{DecF}_{||\cdot||}^d(F) \cdot \text{DecF}_{||\cdot||}^d(G).$$

and the same for permutations.

The $||\cdot||_a$ norm defined in [21] has the quite interesting property that it characterizes the best advantage of a distinguisher in Cl_a^d .

Lemma 1 ([21]). *For any random functions F and G we have*

$$|[F]^d - [G]^d|_a = 2 \cdot \text{Adv}_{\text{Cl}_a^d}(F, G).$$

In this paper, we will use the $\|\cdot\|_a$ norm only and omit it in the notations.

Finally we recall the following lemma.

Lemma 2 ([21]). *Let d be an integer, F_1, \dots, F_r be r random function oracles, and C_1, \dots, C_s be s random permutation oracles. We let Ω be a deterministic oracle Turing machine which can access to the previous oracles and an input tape x . It defines a random function $G(x) = \Omega(F_1, \dots, F_r, C_1, \dots, C_s)(x)$. We assume that Ω is such that the number of queries to F_i is limited to some integer a_i , and the number of queries to C_j is limited to b_j in total for any $i = 1, \dots, r$ and any $j = 1, \dots, s$. We let the F_i^* (resp. C_j^*) be independent uniformly distributed random functions (resp. permutations) on the same range than F_i (resp. C_j) and we let $G^* = \Omega(F_1^*, \dots, F_r^*, C_1^*, \dots, C_s^*)$. We have*

$$\text{DecF}^d(G) \leq \sum_{i=1}^r \text{DecF}^{a_i d}(F_i) + \sum_{j=1}^s \text{DecP}^{b_j d}(C_j) + \text{DecF}^d(G^*).$$

This lemma actually separates the problem of studying the decorrelation bias of a construction scheme into the problem of studying the decorrelation biases of its internal functions F_i and C_j and studying the decorrelation bias of an idealized version G^* .

1.3 The Coefficient H Method

Patarin introduced the “coefficient H method” which enables to make pseudo-randomness proofs more systematic. In the decorrelation theory setting, this method can be formalized by the following lemma.

Lemma 3 ([22]). *Let d be an integer. Let F be a random function from a set \mathcal{M}_1 to a set \mathcal{M}_2 . We let \mathcal{X} be the subset of \mathcal{M}_1^d of all (x_1, \dots, x_d) with pairwise different entries. We let F^* be a uniformly distributed random function from \mathcal{M}_1 to \mathcal{M}_2 . We assume there exist a subset $\mathcal{Y} \subseteq \mathcal{M}_2^d$ and two positive numbers ϵ_1 and ϵ_2 such that*

$$\begin{aligned} & - |\mathcal{Y}|(\#\mathcal{M}_2)^{-d} \geq 1 - \epsilon_1 \\ & - \forall x \in \mathcal{X} \quad \forall y \in \mathcal{Y} \quad [F]_{x,y}^d \geq (1 - \epsilon_2)(\#\mathcal{M}_2)^{-d}. \end{aligned}$$

Then we have $\text{DecF}^d(F) \leq 2\epsilon_1 + 2\epsilon_2$.

This lemma intuitively means that if $[F]_{x,y}^d$ is close to $[F^*]_{x,y}^d$ for all x and almost all y , then the decorrelation bias of F is small. It is quite straightforward with techniques inspired by Patarin [14,15] and Maurer [11].

As an illustration, Lemma 3 can be used in order to prove the famous Luby-Rackoff Theorem easily as shown in Appendix.

Theorem 4 (Luby-Rackoff 1986 [9]). *Let F_1^*, F_2^*, F_3^* be three independent random functions on $\{0, 1\}^{\frac{m}{2}}$ with uniform distribution. We have*

$$\begin{aligned} \text{DecF}^d(\Psi(F_1^*, F_2^*, F_3^*)) & \leq 2d^2 \cdot 2^{-\frac{m}{2}} \\ \text{DecP}^d(\Psi(F_1^*, F_2^*, F_3^*)) & \leq 2d^2 \cdot 2^{-\frac{m}{2}}. \end{aligned}$$

The results hold for Feistel schemes defined from any (quasi)group operation¹.

2 Decorrelation Biases of Functions over an Infinite Domain

In order to define decorrelation biases of MACs, we need to address the problem of having infinite sets. Let for instance F be a random function defined from \mathcal{M}_1^* to \mathcal{M}_2 (\mathcal{M}_1^* is the set of all finite sequences with entries in \mathcal{M}_1). We define the $[F]^{q_1, \dots, q_d}$ matrix with rows defined on $\mathcal{M}_1^{q_1} \times \dots \times \mathcal{M}_1^{q_d}$ and columns defined on \mathcal{M}_2^d . Next we define $\text{DecF}^{q_1, \dots, q_d}(F)$ as the distance between $[F]^{q_1, \dots, q_d}$ and $[F^*]^{q_1, \dots, q_d}$, where F^* has a uniform distribution. Additionally, we can define

$$\text{DecF}^{d,q}(F) = \max_{q_1 + \dots + q_d = q} \text{DecF}^{q_1, \dots, q_d}(F).$$

We can easily check that all previous results remain valid for these definitions, namely:

- The best advantage of a distinguisher limited to d (adaptively) chosen queries with a total length of q blocks between F and F^* is $\frac{1}{2} \text{DecF}^{d,q}(F)$.
- As in Lemma 2, if $G = \Omega(F_1, \dots, F_r, F'_1, \dots, F'_s)$ uses functions F_i and F'_j on fixed input length, but with occurrence numbers of $a_i \ell$ and b_j respectively where ℓ is the length of the input of G , we have

$$\text{DecF}^{d,q}(G) \leq \sum_{i=1}^r \text{DecF}^{a_i q}(F_i) + \sum_{j=1}^s \text{DecF}^{b_j d}(F'_j) + \text{DecF}^{d,q}(G^*).$$

We can use permutations C_i and C'_j as well and have DecP instead of DecF, or even mixtures of functions and permutations.

- Lemma 3 still holds with $\text{DecF}^{d,q}$ instead of DecF^d and \mathcal{X} equal to the set of (x_1, \dots, x_d) with total length q .

3 Security of MAC

Message Authentication Codes (MAC) are functions which map any binary string onto a fixed length value² with a secret key. In this paper, we consider functions defined on the set $(\{0, 1\}^m)^*$ of finite sequences of m -bit integers³. For

¹ Here $\Psi(F_1^*, F_2^*, F_3^*)$ is the standard notation for a Feistel cipher with three rounds and round functions F_1^*, F_2^*, F_3^*

² More precisely, the MAC is the output of the function, but we will improperly call the function a MAC

³ Note that arbitrary bit strings do not always have an integral number of blocks. For this we must use a padding scheme like the Merkle-Damgård [8,13] one in order to transform an arbitrary string into a string with an integral number of blocks. In this paper we prove the security for padded messages which induces the security for the whole scheme with the padding scheme

instance, given a block cipher Enc_K which is a permutation on $\{0, 1\}^m$ defined from a secret key K , we consider the CBC-MAC construction defined by

$$\text{MAC}_K(m_1, \dots, m_\ell) = \text{Enc}_K(\text{Enc}_K(\dots \text{Enc}_K(m_1) + m_2 \dots) + m_\ell).$$

Since the secret key K is unknown by the opponent and chosen at random by the legitimate user, we can consider equivalently $C = \text{Enc}_K$ as a random permutation with a given publicly known distribution, and the MAC itself as a random function.

The purpose of MACs is to authenticate messages. Namely, the legitimate authenticator provides $\text{MAC}(x)$ in order to authenticate a message x . Saying that a MAC is (d, q, p) -secure means that for any opponent who can use the legitimate authenticator as an oracle for at most $d - 1$ chosen messages x_1, \dots, x_{d-1} and issue an (x_d, c) pair such that $x_d \neq x_i$ for any i and that the total length of x_1, \dots, x_d is of q m -bit blocks, the probability that $c = \text{MAC}(x_d)$ is less than p . This is the security against adaptive existential forgery attacks.

We notice that if MAC is such that $\text{DecF}^{d,q}(\text{MAC}) = \epsilon$, then it is a $(d, q, 2^{-m} + \frac{\epsilon}{2})$ -secure MAC. Namely, for any opponent we can make a distinguisher who just query the forged x_d and check whether the output is c or not. Since the advantage must be less than $\frac{\epsilon}{2}$, the probability of success of the opponent must be less than $\frac{\epsilon}{2}$ plus the probability of success against a truly random function, which is 2^{-m} . Hence we use $\text{DecF}^{d,q}(\text{MAC})$ upper bounds as security evidences.

For instance, we can consider the Bellare-Kilian-Rogaway result which works with a fixed input length ℓ .

Theorem 5 (Bellare-Kilian-Rogaway 1994 [6]). *For any fixed integer ℓ , we consider the function MAC defined on ℓ m -bit blocks from a uniformly distributed random function F^* as follows.*

$$\text{MAC}(m_1, \dots, m_\ell) = F^*(F^*(\dots F^*(m_1) + m_2 \dots) + m_\ell).$$

For any d we have $\text{DecF}^d(\text{MAC}) \leq 6d^2\ell^2 2^{-m}$. This holds for any (quasi)group addition.

Here is another result which is quite similar to the An-Bellare result [5].

Theorem 6 ([22]). *Let F_1 and F_2 be two independent random functions from $\{0, 1\}^{b+m}$ to $\{0, 1\}^b$. For any ℓ and any $(m_1, \dots, m_\ell) \in (\{0, 1\}^m)^\ell$ we define*

$$\text{MAC}(m_1, \dots, m_\ell) = F_2(F_1(\dots F_1(F_1(0, m_1), m_2) \dots, m_\ell), \ell)$$

where 0 means a b -bit zero string, and ℓ means an m -bit string which represents the ℓ value. Considering distinguishers limited to d queries and a total length of qm bits we have

$$\text{DecF}^{d,q} \leq \text{DecF}^q(F_1) + \text{DecF}^d(F_2) + q(q-1)2^{-m}.$$

Finally, here is the Petrank-Rackoff [16] result.

Theorem 7 (Petrank-Rackoff [16]). *Let C_1 and C_2 be two independent random permutations on $\{0,1\}^m$ with the same distribution C . For any ℓ and any $(m_1, \dots, m_\ell) \in (\{0,1\}^m)^\ell$ we define*

$$\text{MAC}(m_1, \dots, m_\ell) = C_2(C_1(C_1(\dots C_1(C_1(m_1) + m_2) \dots + m_{\ell-1}) + m_\ell)).$$

Considering adaptive distinguishers limited to d queries and a total length of qm bits we have

$$\text{DecF}^{d,q}(\text{MAC}) \leq 2\text{DecP}^q(C) + 4q^2 2^{-m}.$$

The result holds for any (quasi)group addition.

4 Encrypted CBC-MAC

Here is our main result.

Theorem 8. *Let C_1 and C_2 be two independent random permutations over $\{0,1\}^m$. For any ℓ and any $(m_1, \dots, m_\ell) \in (\{0,1\}^m)^\ell$ we define*

$$\text{MAC}(m_1, \dots, m_\ell) = C_2(C_1(\dots C_1(C_1(m_1) + m_2) \dots + m_{\ell-1}) + m_\ell).$$

Considering adaptive distinguishers limited to d queries and a total length of qm bits we have

$$\begin{aligned} \text{DecF}^{d,q}(\text{MAC}) &\leq \text{DecP}^{q-d}(C_1) + \text{DecP}^d(C_2) \\ &\quad + d(d-1)2^{-m} + q(q+1)(1+q2^{-m})2^{-m}. \end{aligned}$$

The result holds for any (quasi)group addition.

This result is slightly better than Theorem 7.

Proof. Lemma 2 reduces to the case where C_1 and C_2 are independent uniformly distributed random permutations.

Using Lemma 3, let \mathcal{Y} be the set of all $y = (y_1, \dots, y_d)$ with different y_i s. We thus have

$$\epsilon_1 = 1 - \frac{2^{md}}{2^m(2^m - 1) \dots (2^m - d + 1)} \leq \frac{d(d-1)}{2} 2^{-m}.$$

Now for any collection of $x_i = (m_{i,1}, \dots, m_{i,q_i})$ we let

$$U_{i,j} = C_1(\dots C_1(C_1(m_{i,1}) + m_{i,2}) \dots + m_{i,j-1}) + m_{i,j}.$$

We consider the event E that all U_{i,q_i} are pairwise different. We have

$$\begin{aligned} [\text{MAC}]_{x,y}^{q_1, \dots, q_d} &\geq \Pr[\text{MAC}(x_i) = y_i; i = 1, \dots, d \text{ and } E] \\ &= \Pr[\text{MAC}(x_i) = y_i; i = 1, \dots, d/E] \Pr[E] \\ &= \frac{1}{2^m(2^m - 1) \dots (2^m - d + 1)} \Pr[E] \\ &\geq 2^{-md}(1 - \Pr[\bar{E}]) \end{aligned}$$

therefore we can take $\epsilon_2 = \Pr[\bar{E}] = \Pr[\exists i < r; U_{i,q_i} = U_{r,q_r}]$.

The remaining part of the proof consists of upper bounding ϵ_2 by $\frac{q(q+1)}{2}(1 + q2^{-m})2^{-m}$ and applying Lemma 3.

We call a collision an event $U_{i,j} = U_{r,s}$. This collision is trivial if we have $(m_{i,1}, \dots, m_{i,j}) = (m_{r,1}, \dots, m_{r,s})$ and non-trivial otherwise. Let Inv be the event that $C_1(U_{i,j}) = 0$ for some i, j , and let Coll be the event that we have a non-trivial collision. We can easily show that the \bar{E} event is included in $\text{Inv} \cup \text{Coll}$: if $U_{i,q_i} = U_{r,q_r}$, then either $m_{i,q_i} \neq m_{r,q_r}$ and it is a non-trivial collision, or it reduces to $U_{i,q_i-1} = U_{r,q_r-1}$ and we can iterate... Thus $\epsilon_2 \leq \Pr[\text{Inv}] + \Pr[\text{Coll}]$.

The probability that any adaptive attack against C_1 finds a preimage of 0 after $q - d$ queries is obviously less than $\frac{q}{2^m - q}$. Thus $\Pr[\text{Inv}] \leq \frac{q}{2^m - q}$.

We let \mathcal{U} be the set of all $U_{i,j}$ -indices, which means the set of all (i, j) such that $1 \leq i \leq d$ and $1 \leq j \leq q_i$. For $A \subseteq \mathcal{U}$ we let $c(A)$ be

$$c(A) = \{(i, j); \exists (r, s) \in A \text{ } i = r \text{ and } j \leq s\}.$$

Thus $c(A)$ is the set the indices of all $U_{i,j}$ which are required in order to compute all $U_{r,s}$ values for $(r, s) \in A$. We define an ordering on $2^{\mathcal{U}}$ by

$$A \leq B \iff c(A) \subseteq c(B).$$

We let \mathcal{I} be the set of all indices pairs of potential non-trivial collisions $U_{i,j} = U_{r,s}$, namely the set of all pairs $\{(i, j), (r, s)\}$ of \mathcal{U} -elements such that $(m_{i,1}, \dots, m_{i,j}) \neq (m_{r,1}, \dots, m_{r,s})$. For any i, j, r, s such that $\{(i, j), (r, s)\} \in \mathcal{I}$ we let $\text{Coll}_{i,j,r,s}$ be the event of the collision $U_{i,j} = U_{r,s}$ (which is necessarily non-trivial since $\{(i, j), (r, s)\} \in \mathcal{I}$), and we let $\text{MinColl}_{i,j,r,s}$ be the complementary in $\text{Coll}_{i,j,r,s}$ of the union of all $\text{Coll}_{i',j',r',s'}$ for $\{(i', j'), (r', s')\} \in \mathcal{I}$ and $\{(i', j'), (r', s')\} < \{(i, j), (r, s)\}$, i.e. the event $U_{i,j} = U_{r,s}$ with no prior non-trivial collision. We easily notice that

$$\text{Coll} = \bigcup_{\{(i,j),(r,s)\} \in \mathcal{I}} \text{MinColl}_{i,j,r,s}.$$

We have at most $\frac{q(q-1)}{2}$ terms in \mathcal{I} . Hence

$$\Pr[\text{Coll}] \leq \frac{q(q-1)}{2} \max_{\{(i,j),(r,s)\} \in \mathcal{I}} \Pr[\text{MinColl}_{i,j,r,s}].$$

For $\{(i, j), (r, s)\} \in \mathcal{I}$, let us consider the $\text{MinColl}_{i,j,r,s}$ event. We assume without loss of generality that $s \leq j$. Since we have no prior collision we must have $m_{i,j} \neq m_{r,s}$. Furthermore we must have $U_{i,j-1} \neq U_{r,s-1}$ because C_1 is a permutation (otherwise $C_1(U_{i,j-1}) + m_{i,j}$ cannot be equal to $C_1(U_{r,s-1}) + m_{r,s}$) and $j > 1$, and we need to consider the event

$$C_1(U_{i,j-1}) + m_{i,j} = U_{r,s}.$$

If we have a collision $U_{i,j-1} = U_{i',j'}$ with $(i, j-1) \neq (i', j')$ and $(i', j') \in c(i, j, r, s)$, it must be trivial (otherwise the initial collision is not minimal) which

means $j' = j - 1$ and $i' = r \neq i$ and $(m_{i,1}, \dots, m_{i,j-1}) = (m_{r,1}, \dots, m_{r,j-1})$. If $s < j$ we have $U_{i,j} = U_{r,s}$ and $U_{r,s} = U_{i,s}$ thus $U_{i,j} = U_{i,s}$ which is non-trivial, which contradicts the minimality of the initial collision. Thus we must have $s = j$, but the trivial collision $U_{i,j-1} = U_{r,j-1}$ then contradicts $U_{i,j-1} \neq U_{r,s-1}$. Therefore $U_{i,j-1}$ is equal to no $U_{i',j'}$ for $(i', j') \in c(i, j, r, s) \setminus \{(i, j - 1)\}$. This implies that the marginal distribution of $C_1(U_{i,j-1})$ with the knowledge of all previous $U_{i',j'}$ is uniform among a set of at least $2^m - q + 1$ elements. Hence $\Pr[\text{MinColl}_{i,j,r,s}] \leq \frac{1}{2^m - q}$.

Finally we obtain

$$\epsilon_2 \leq \frac{q}{2^m - q} + \frac{q(q-1)}{2} \times \frac{1}{2^m - q} \leq \frac{q(q+1)}{2} (1 + q2^{-m})2^{-m}.$$

Applying Lemma 3 now completes the proof. \square

5 Extensions

In our result we notice that since $d \leq q$, the bound is small until q reaches the order of $2^{\frac{m}{2}}$. This result is tight since usual collision attacks can break our construction within this complexity. Actually, we can query $2^{\frac{m}{2}}$ two-block messages until we get a collision $\text{MAC}(m_1, m_2) = \text{MAC}(m'_1, m'_2)$ then query $c = \text{MAC}(m_1, m_2, m_3)$ and output a forged authenticated message $((m'_1, m'_2, m_3), c)$. We have $d = 2^{\frac{m}{2}} + 2$ and $q = 2 \cdot 2^{\frac{m}{2}} + 6$ and $p \approx 1 - e^{-1}$.

We may think that since we have an m -bit MAC and a security of $2^{\frac{m}{2}}$ uses we have an efficiency loss in term of storage. We can improve this construction by shrinking the MAC on $\frac{m}{2}$ bits as suggested in most of standards. More precisely, let F be a random function from $\{0, 1\}^m$ to $\{0, 1\}^b$. We can define

$$\text{MAC}(m_1, \dots, m_\ell) = F(C(\dots C(C(m_1) + m_2) \dots + m_{\ell-1}) + m_\ell)$$

and we have

$$\text{DecF}^{d,q}(\text{MAC}) \leq \text{DecP}^q(C) + \text{DecF}^d(F) + q(q+1)(1 + q2^{-m})2^{-m}.$$

(In the proof of Theorem 8, we take \mathcal{Y} equal to the full set so that $\epsilon_1 = 0$.)

If we now want to shorten the two keys, we can replace the independent C and F random functions by dependent ones. Let $\| [CF]^q - [C_0 F_0]^q \|_a$ denote the decorrelation distance between the (C, F) pair and a pair (C_0, F_0) of independent random functions such that C_0 (resp. F_0) has the same distribution than C (resp. F). This is half of the best advantage for distinguishing them from q queries. We should still consider $\text{DecP}^{q-d}(C)$ and $\text{DecF}^d(F)$. So, even if C and F are dependent, we still have the following result.

Theorem 9. *Let C and C_0 be two identically distributed random permutations on $\{0, 1\}^m$ and let F and F_0 be two identically distributed random functions from $\{0, 1\}^m$ to $\{0, 1\}^b$. We assume that C_0 and F_0 are independent. For any ℓ and any $(m_1, \dots, m_\ell) \in (\{0, 1\}^m)^\ell$ we define*

$$\text{MAC}(m_1, \dots, m_\ell) = F(C(\dots C(C(m_1) + m_2) \dots + m_{\ell-1}) + m_\ell).$$

Considering adaptive distinguishers limited to d queries and a total length of qm bits we have

$$\text{DecF}^{d,q}(\text{MAC}) \leq ||[CF]^q - [C_0F_0]^q||_a + \text{DecP}^{q-d}(C) + \text{DecF}^d(F) + q(q+1)(1+q2^{-m})2^{-m}.$$

The result holds for any (quasi)group addition.

This theorem clearly separates the security issues induced by the probabilistic dependence between C and F , the C algorithm, the F algorithm, and the MAC scheme.

As an example we can use

$$C(x) = \text{DES}_K(x) \quad \text{and} \quad F(x) = \text{Trunc}(\text{DES}_{K+c}(x))$$

for a given constant c , and where Trunc truncates a 64-bit string onto its first half and DES is the Data Encryption Standard [1]. We get a MAC on $b = 32$ bits with a single 56-bit key and block of $m = 64$ bits. We obtain

$$\text{DecF}^{d,q}(\text{MAC}) \leq f(q) + q(q+1)(1+q2^{-64})2^{-64}$$

where $f(q)$ is the sum of the best advantages for distinguishing

- $(\text{DES}_K, \text{Trunc} \circ \text{DES}_{K+c})$ from $(\text{DES}_{K_1}, \text{Trunc} \circ \text{DES}_{K_2})$
- DES from C^*
- $\text{Trunc} \circ \text{DES}$ from F^*

within a total number of query blocks less than q . Let $q = \theta 2^{\frac{m}{2}}$ (which is a limit of 320GB of queries). The advantage of any distinguisher is less than $\frac{f(q)+\theta^2}{2}$ thus the probability of success of any adaptive existential forgery attack is less than $2^{-32} + \frac{f(q)+\theta^2}{2}$. Let us conjecture that $f\left(\frac{2^{32}}{10}\right) \leq 2^{-7}$. If we authenticate less than 3GB, the probability of success of the best attack is less than 1%.

The Advanced Encryption Standard will soon provide better security with $m = 128$.

It shall however be outlined that this example is a little misleading since we do not assume any computational bound on the distinguisher which can thus perform an exhaustive search. This means that the conjecture is wrong. We can still modify the result and the computational model by limiting the time complexity to t . All reductions in this paper introduce simulators (like for instance a simulator for the MAC given an oracle for DES) which induce a small time complexity overhead which is often denoted $O(1)$. As a result we obtain that for the maximal time complexity t such that $f\left(\frac{2^{32}}{10}\right) \leq 2^{-7}$, the probability of success of any attack which is limited to a complexity of $t - O(1)$ is less than 1% after having authenticated 3GB.

6 Conclusion

We have shown that the regular CBC-MAC construction provides a secure MAC when the output is encrypted. The security analysis suggests that if m is the block length of the underlying block cipher, then we should not use the MAC construction on more than $2^{\frac{m}{2}}$ blocks in total.

In order to fit to the security, we can even reduce the MAC length down to $\frac{m}{2}$ bits, and shorten the key with extra security hypothesis. This enables to prove the security of existing standards.

These results are quite similar than the Petrank-Rackoff ones. Our technique based on decorrelation theory is however quite systematic and can be applied to most of current MAC constructions with compact proofs.

Finally, we believe that these techniques will contribute to making systematic proof analysis of cryptographic schemes and ultimately lead to some automatic security validation procedures.

A Proof of Theorem 4

Following the Feistel scheme $F = \Psi(F_1^*, F_2^*, F_3^*)$, we let

$$\begin{aligned} x_i &= (z_i^0, z_i^1) \\ z_i^2 &= z_i^0 + F_1^*(z_i^1) \\ y_i &= (z_i^4, z_i^3) \end{aligned}$$

We let E be the event $z_i^3 = z_i^1 + F_2^*(z_i^2)$ and $z_i^4 = z_i^2 + F_3^*(z_i^3)$ for $i = 1, \dots, d$. We thus have $[F]_{x,y}^d = \Pr[E]$. We now define

$$\mathcal{Y} = \{(y_1, \dots, y_d); \forall i < j \quad z_i^3 \neq z_j^3\}.$$

We can easily check that \mathcal{Y} fulfill the requirements of Lemma 3. Firstly we have

$$|\mathcal{Y}| \geq \left(1 - \frac{d(d-1)}{2} 2^{-\frac{m}{2}}\right) 2^{md}$$

thus we let $\epsilon_1 = \frac{d(d-1)}{2} 2^{-\frac{m}{2}}$. Second, for $y \in \mathcal{Y}$ and any x (with pairwise different entries), we need to consider $[F]_{x,y}^d$. Let E^2 be the event that all z_i^2 s are pairwise different over the distribution of F_1^* . We have

$$[F]_{x,y}^d \geq \Pr[E/E^2] \Pr[E^2].$$

For computing $\Pr[E/E^2]$ we know that z_i^3 s are pairwise different, as for the z_i^2 s. Hence $\Pr[E/E^2] = 2^{-md}$. It is then straightforward that $\Pr[E^2] \geq 1 - \frac{d(d-1)}{2} 2^{-\frac{m}{2}}$ which is $1 - \epsilon_2$. We thus obtain from Lemma 3 that $\text{DecF}^d(F) \leq 2d(d-1)2^{-\frac{m}{2}}$. From Lemma 3 it is straightforward that $\text{DecF}^d(C^*) \leq d(d-1)2^{-m}$. We thus obtain $\text{DecP}^d(F) \leq 2d^2 2^{-\frac{m}{2}}$ for $d \leq 2^{1+\frac{m}{2}}$. Since DecF is always less than 2, it also holds for larger d . \square

References

1. Data Encryption Standard. *Federal Information Processing Standard Publication 46*, U. S. National Bureau of Standards, 1977.
2. ANSI X9.9. American National Standard - Financial Institution Message Authentication (Wholesale). ASC X9 Secretariat - American Bankers Association, 1986.
3. ISO 8731-2. Banking - Approved Algorithms for Message Authentication - Part 2: Message Authenticator Algorithm. International Organization for Standardization, Geneva, Switzerland, 1992.
4. *RACE Project*, Lectures Notes in Computer Science 1005, Springer-Verlag, 1995. .
5. J. H. An, M. Bellare. Constructing VIL-MACs from FIL-MACs: Message Authentication under Weakened Assumptions. In *Advances in Cryptology CRYPTO'99*, Santa Barbara, California, U.S.A., Lectures Notes in Computer Science 1666, pp. 252–269, Springer-Verlag, 1999.
6. M. Bellare, J. Kilian, P. Rogaway. The Security of Cipher Block Chaining. In *Advances in Cryptology CRYPTO'94*, Santa Barbara, California, U.S.A., Lectures Notes in Computer Science 839, pp. 341–358, Springer-Verlag, 1994.
7. E. Biham, A. Shamir. *Differential Cryptanalysis of the Data Encryption Standard*, Springer-Verlag, 1993.
8. I. B. Damgård. A Design Principle for Hash Functions. In *Advances in Cryptology CRYPTO'89*, Santa Barbara, California, U.S.A., Lectures Notes in Computer Science 435, pp. 416–427, Springer-Verlag, 1990.
9. M. Luby, C. Rackoff. How to Construct Pseudorandom Permutations from Pseudorandom Functions. *SIAM Journal on Computing*, vol. 17, pp. 373–386, 1988.
10. M. Matsui. The first experimental cryptanalysis of the Data Encryption Standard. In *Advances in Cryptology CRYPTO'94*, Santa Barbara, California, U.S.A., Lectures Notes in Computer Science 839, pp. 1–11, Springer-Verlag, 1994.
11. U. M. Maurer. A Simplified and Generalized Treatment of Luby-Rackoff Pseudorandom permutation generators. In *Advances in Cryptology EUROCRYPT'92*, Balatonfüred, Hungary, Lectures Notes in Computer Science 658, pp. 239–255, Springer-Verlag, 1993.
12. U. M. Maurer. Information-Theoretic Cryptography. Invited lecture. In *Advances in Cryptology CRYPTO'99*, Santa Barbara, California, U.S.A., Lectures Notes in Computer Science 1666, pp. 47–64, Springer-Verlag, 1999.
13. R. C. Merkle. One way Hash Functions and DES. In *Advances in Cryptology CRYPTO'89*, Santa Barbara, California, U.S.A., Lectures Notes in Computer Science 435, pp. 416–427, Springer-Verlag, 1990.
14. J. Patarin. *Etude des Générateurs de Permutations Basés sur le Schéma du D.E.S.*, Thèse de Doctorat de l'Université de Paris 6, 1991.
15. J. Patarin. How to Construct Pseudorandom and Super Pseudorandom Permutations from One Single Pseudorandom Function. In *Advances in Cryptology EUROCRYPT'92*, Balatonfüred, Hungary, Lectures Notes in Computer Science 658, pp. 256–266, Springer-Verlag, 1993.
16. E. Petrank, C. Rackoff. CBC MAC for Real-Time Data Sources. *Journal of Cryptology*, vol. 13, pp. 315–338, 2000.
17. S. Vaudenay. Provable Security for Block Ciphers by Decorrelation. Invited talk. In *STACS 98*, Paris, France, Lectures Notes in Computer Science 1373, pp. 249–275, Springer-Verlag, 1998. Full Paper: technical report LIENS-98-8, Ecole Normale Supérieure, 1998. (<ftp://ftp.ens.fr/pub/reports/liens/>)

18. S. Vaudenay. Feistel Ciphers with L_2 -Decorrelation. In *Selected Areas in Cryptography*, Kingston, Ontario, Canada, Lectures Notes in Computer Science 1556, pp. 1–14, Springer-Verlag, 1999.
19. S. Vaudenay. Resistance Against General Iterated Attacks. In *Advances in Cryptology EUROCRYPT'99*, Prague, Czech Republic, Lectures Notes in Computer Science 1592, pp. 255–271, Springer-Verlag, 1999.
20. S. Vaudenay. On the Lai-Massey Scheme.
21. S. Vaudenay. Adaptive-Attack Norm for Decorrelation and Super-Pseudorandomness. In *Selected Areas in Cryptography*, Kingston, Ontario, Canada, Lectures Notes in Computer Science 1758, pp. 49–61, Springer-Verlag, 2000.
22. S. Vaudenay. On Provable Security for Conventional Cryptography. Invited talk. (To appear in the proceedings of ICISC' 99, LNCS, Springer-Verlag.)

HAS-V: A New Hash Function with Variable Output Length

Nan Kyoung Park*, Joon Ho Hwang, and Pil Joong Lee

Pohang University of Science and Technology(POSTECH),
Pohang, 790-784, Korea
nkpark@mail.posdata.co.kr, jhhwang@oberon.postech.ac.kr,
pjl@postech.ac.kr

Abstract. Hash functions play an essential role in many areas of cryptographic applications such as digital signature, authentication, and key derivation. In this paper, we propose a new hash function with variable output length, namely HAS-V, to meet the needs of various security levels desired among different applications. A great deal of attention was paid to balance the characteristics of security and performance. The use of message expansion, 4-variable Boolean functions, variable and fixed amounts of shifts, and interrelated parallel lines provide a high level of security for HAS-V. Experiments show that HAS-V is about 19% faster than SHA-1, 31% faster than RIPEMD-160, and 26% faster than HAVAL on a Pentium PC.

1 Introduction

A hash function is a function that maps an input with an arbitrary length to an output with a specific length, referred to as a *hash-code*. A *one-way* hash function must obey the preimage and second preimage resistance properties. Furthermore, most cryptographic applications require the hash function to satisfy the collision resistance property, which is a stronger constraint than the former two properties.

The collision of a hash function can be found by the birthday paradox or square root attack with $2^{n/2}$ operations where n is the length of the hash-code [18]. In order to prevent such attacks, the length of the hash-code should be no less than 128 bits. However, the works of van Oorschot and Wiener [12], on special-purpose hardware design for parallel collision search, suggest that the minimum length of the hash-code should be 160 bits. Ever since Damgård [6] established the design principles of a hash function, which included the fact that the collision resistance of the compression function is sufficient for the collision resistance of the hash function, almost all hash functions follow these principles.

There are three main categories of hash functions, namely hash functions based on block ciphers, hash functions based on modular arithmetic, and dedicated hash functions [13]. Most early hash functions were based on block ciphers. However, the modification of block ciphers into hash functions resulted

* Nan Kyoung Park is now with POSDATA, Pun-Dang Gu, Sung-Nam, 463-050, Korea

in security weaknesses in collision search as well as performance deterioration. The slow performance problem is even more serious for hash functions based on modular arithmetic and serious doubts have been raised about their security. Consequently, the need for fast and secure hash functions resulted in dedicated hash functions. These functions are custom designed to achieve the goals of a cryptographic hash function. Among the numerous dedicated hash functions that are in use today, the MD4-family hash functions are the most widely used and analyzed family of hash functions. MD4 [14], MD5 [15], and RIPEMD-160 [9] are popular examples of MD4-family hash functions.

There have been several attacks on the MD4-family hash functions [1,2,5,7]. Among these attacks, a series of Dobbertin's attacks are becoming a real threat on practical applications. Fortunately, SHA-1 [11] and RIPEMD-160 are considered to be secure against these attacks [8]. The main distinction of SHA-1 is the message expansion process, where the message words used in the different rounds are computed as the sum of the previous message words and circular shift by 1-bit. This prevents making local changes, which is confined to a few bits, and accordingly individual message bits influence the calculations at large number of places. RIPEMD-160 is an enhanced version, in a way to be resistant against Dobbertin's attacks, of RIPEMD. Its main improvements are the increase in the number of rounds from 3 to 5 and the two parallel lines were modified to have a different message ordering, Boolean functions, and shift amounts.

Recently, much progress has been made in the software implementation of MD4-family hash functions [3,4]. Analyses show that the structures of MD4-family hash functions possess a higher instruction-level parallelism than current general-purpose computer architecture can provide. Among the MD4-family hash functions, it is known that the critical path to compute the step function of SHA-1 is shorter than any other MD4-family hash functions and the organization of RIPEMD-160 in two independent lines will become much useful in the near future.

In the remainder of this paper, we propose a new MD4-family hash function that produces a variable length hash-code, namely HAS-V. In Section 2, we present details on why a hash function with variable length hash-code is needed. In Section 3, the terminologies and notations used are defined. In Section 4, a description of the newly proposed hash function is given. In Section 5, we discuss the underlying design principles of HAS-V based on performance and security aspects. Performance comparison is given in Section 6, and concluding remarks are given in Section 7. The pseudo-code and the test values of HAS-V are given in the Appendix.

2 Motivation

The length of the hash-code is an important factor directly connected to the security of the hash function. Assuming that there are no unexpected design flaws known in a hash function, the complexity of finding a collision is heavily dependent on the length of the hash-code. The length of the hash-code must

be long enough to provide explicit security, but it must not be unnecessarily long to sacrifice efficiency of the entire system. Consequently, the length of the hash-code is a relative factor to the computing power possessed by the opponent. Therefore, the length of the hash-code is meant to vary from time to time as technology advances and information security broadens its area of application. It seems inappropriate to fix the length of the hash-code when various levels of security are desired among different applications.

KCDSA [10], Korea Certificate-based Digital Signature Algorithm, is an example of a cryptographic application where a variable length of hash-code is needed. KCDSA employs variable length domain parameters in order to fulfill the various security needs in different applications. In the case of KCDSA, there is a need for a hash function that can produce a variable hash-code of up to 256 bits in order to fully utilize the flexible security level of KCDSA.

However, most conventional hash algorithms are designed to produce a specific length of hash-code, such as 128 bits for MD4 and MD5, and 160 bits for SHA-1 and RIPEMD-160. Among the well-known hash functions, HAVAL [19] is the only hash function that can produce a variable length hash-code. Although HAVAL is still considered to be secure, there are some concerns that a suitable modification of MD4 attack could be applied to HAVAL with 3 passes. Furthermore, HAVAL suffers from performance deterioration in CISC processors due to the excessive number of chaining variables used. There exists an optional extension of RIPEMD-128 and RIPEMD-160 to produce 256-bit and 320-bit hash-code. However, these methods do not provide any increase in security level, merely an increase in the length of the hash-code. This gives a clear motivation to design a new hash function with variable length hash-code, which is both efficient and secure.

Information security is becoming an inevitable part of our society, and therefore information technology must provide services to fulfill the needs of various people. The need for variable length hash-code can be explained in an analogous way. Moreover, the ever-increasing nature of computing power will eventually threaten the length of the hash-codes used in many applications today. Instead of redesigning a new hash function in such events, the use of a single hash function with a variable length hash-code seems to be a cost-effective and convenient way of increasing the security level.

3 Terminology and Notations

The use of *byte* in this paper implies an 8-bit quantity, *word* implies a 32-bit quantity, and *block* implies a 1024-bit quantity, which is the input size of the compression function. We assume a byte with the most significant bit of each byte listed first and a block with the least significant byte of each block given first. Throughout this paper, the following notations will be used:

- $+$: addition of words, i.e. addition by modulo- 2^{32} .
- $X \ll^s$: the circular left shift of X by s bit positions.

Table 1. Characteristics of HAS-V

| | |
|------------------------------|-----------|
| Length of Input Block (bits) | 1024 |
| Length of Output (bits) | 128 ~ 320 |
| Number of Rounds | 10 |
| Number of Chaining Variables | 10 |
| Number of Steps | 200 |

Table 2. Initial values

| | | | | |
|----------|-----------------|-----------------|-----------------|-----------------|
| <i>A</i> | <i>B</i> | <i>C</i> | <i>D</i> | <i>E</i> |
| 67452301 | <i>efcdab89</i> | <i>98badcfe</i> | 10325476 | <i>c3d2e1f0</i> |
| <i>F</i> | <i>G</i> | <i>H</i> | <i>I</i> | <i>J</i> |
| 8796a5b4 | <i>4b5a6978</i> | <i>0f1e2d3c</i> | <i>a0b1c2d3</i> | <i>68794e5f</i> |

- \neg : the bitwise complement operation.
- \vee, \wedge, \oplus : the bitwise OR, AND, and XOR operation. ($X \wedge Y$ is also denoted as XY for simplicity)

4 Description of HAS-V Algorithm

The basic structure of the compression function of HAS-V is two parallel lines, denoted as the X-line and the Y-line, consisting of 100 steps each. Each line is composed of 5 rounds, where each round consists of 20 steps, and maintains 5 words of chaining variables, a total of 10 chaining variables for the entire compression function. The two parallel lines are interrelated by swapping the contents of the entire chaining variables in the X-line and the Y-line after each round. The message words used in the compression function are 32 words, or a 1024 bit block, of the input message and 8 additionally generated words each round by message expansion, a total of 40 words for the entire compression function. The characteristics of the structure of HAS-V are summarized in Table 1.

Append Padding Bits and Length: The message is padded so that its length is congruent to 952 modulo 1024. Padding is performed by appending a single "1" bit and necessary zero bits to satisfy the above constraints. The remaining 72 bits, in order to be a multiple of 1024 bits, is filled by appending the desired length of the hash-code represented in *bytes* and the length of the input coded in modulo 2^{64} represented in *bits*.

Initial Value of the Chaining Variables: The initial values of the chaining variables used in HAS-V are given in Table 2. *A, B, C, D, E* are the chaining variables of the X-line and *F, G, H, I, J* are the chaining variables of the Y-line.

Message Preparation and Expansion: The length of the input block used in each compression function is 1024 bits. The upper 512 bits consist of 16 words,

Table 3. Message combination to generate extra messages

| Index | Round 1 | Round 2 | Round 3 | Round 4 | Round 5 |
|-------|----------------|--------------|--------------|--------------|--------------|
| 16 | 0, 1, 2, 3 | 3, 6, 9, 12 | 12, 5, 14, 7 | 7, 2, 13, 8 | 15, 9, 5, 3 |
| 17 | 4, 5, 6, 7 | 15, 2, 5, 8 | 0, 9, 2, 11 | 3, 14, 9, 4 | 12, 8, 6, 2 |
| 18 | 8, 9, 10, 11 | 11, 14, 1, 4 | 4, 13, 6, 15 | 15, 10, 5, 0 | 13, 11, 7, 1 |
| 19 | 12, 13, 14, 15 | 7, 10, 13, 0 | 8, 1, 10, 3 | 11, 6, 1, 12 | 14, 10, 4, 0 |

$X[0], X[1], \dots, X[15]$ that are used in the X-line and the remaining lower 512 bits consist of 16 words, $Y[0], Y[1], \dots, Y[15]$ that are used in the Y-line. Each line of the compression function then additionally generates 4 message words in each round by message expansion. The extra messages are created by the XOR of 4-word combinations chosen from its present line. The word combinations used in message expansion differ for every round of the compression function and are given in Table 3.

For example, the message word $X[17]$ used in round 2 of the X-line is generated as follows:

$$X[17] = X[15] \oplus X[2] \oplus X[5] \oplus X[8].$$

The rest of the message words, $X[16], X[18], X[19]$, can be expanded in a similar way. The message words in the opposite Y-line, $Y[16], Y[17], Y[18], Y[19]$ can be derived in an analogous way.

Ordering of the Message Words: Each message word among the 20 message words, 16 initial input message words and 4 expanded message words, is applied to a single step in each line. The order of message words used in both lines is equivalent. The ordering of the message words for each round is given in Table 4. The extra messages generated by message expansion, $X[16], X[17], X[18], X[19]$, are applied to steps 10, 15, 0, and 5, respectively, in each round.

Step Operation: The operation in each step is equivalent in both the X-line and the Y-line. The step operation of the X-line is given below.

$$T \leftarrow A^{\ll s} + f(B, C, D, E) + X + K,$$

$$E \leftarrow D ; D \leftarrow C ; C \leftarrow B^{\ll 30} ; B \leftarrow A ; A \leftarrow T.$$

Here f , s , and K are the Boolean function, shift amount, and additive constant, respectively. The Boolean function and constant differ for every round of the compression function, whereas, the shift amount varies for every step within a single round of the compression function.

Boolean Function: The following 5 Boolean functions are used in HAS-V.

$$f_0(x, y, z, u) = xy \oplus \neg xz \oplus yu \oplus zu,$$

Table 4. Message ordering

| Step | Round 1 | Round 2 | Round 3 | Round 4 | Round 5 |
|------|---------|---------|---------|---------|---------|
| 0 | 18 | 18 | 18 | 18 | 18 |
| 1 | 0 | 3 | 12 | 7 | 15 |
| 2 | 1 | 6 | 5 | 2 | 9 |
| 3 | 2 | 9 | 14 | 13 | 5 |
| 4 | 3 | 12 | 7 | 8 | 3 |
| 5 | 19 | 19 | 19 | 19 | 19 |
| 6 | 4 | 15 | 0 | 3 | 12 |
| 7 | 5 | 2 | 9 | 14 | 8 |
| 8 | 6 | 5 | 2 | 9 | 6 |
| 9 | 7 | 8 | 11 | 4 | 2 |
| 10 | 16 | 16 | 16 | 16 | 16 |
| 11 | 8 | 11 | 4 | 15 | 13 |
| 12 | 9 | 14 | 13 | 10 | 11 |
| 13 | 10 | 1 | 6 | 5 | 7 |
| 14 | 11 | 4 | 15 | 0 | 1 |
| 15 | 17 | 17 | 17 | 17 | 17 |
| 16 | 12 | 7 | 8 | 11 | 14 |
| 17 | 13 | 10 | 1 | 6 | 10 |
| 18 | 14 | 13 | 10 | 1 | 4 |
| 19 | 15 | 0 | 3 | 12 | 0 |

Table 5. Order of Boolean function

| Line | Round 1 | Round 2 | Round 3 | Round 4 | Round 5 |
|------|---------|---------|---------|---------|---------|
| X | f_0 | f_1 | f_2 | f_3 | f_4 |
| Y | f_4 | f_3 | f_2 | f_1 | f_0 |

$$\begin{aligned}
f_1(x, y, z, u) &= xz \oplus y \oplus u, \\
f_2(x, y, z, u) &= xy \oplus \neg xu \oplus z, \\
f_3(x, y, z, u) &= x \oplus yz \oplus u (= f_1(y, x, z, u)), \\
f_4(x, y, z, u) &= \neg xy \oplus xz \oplus yu \oplus zu (= f_0(x, z, y, u)).
\end{aligned}$$

The Boolean functions are applied, in each line, as in Table 5

Shifts: For both lines, the shift amount is given in Table 6. The period of the shift amount is 20 steps in the compression function.

Constants: Additive constants are taken as the integer parts of the numbers given in Table 7.

Swapping of the Chaining Variables: The contents of the chaining variables in the X-line and the Y-line are swapped after every round.

$$A \leftrightarrow F ; B \leftrightarrow G ; C \leftrightarrow H ; D \leftrightarrow I ; E \leftrightarrow J$$

Table 6. Shift amount

| | | | | | | | | | | |
|-------------|----|----|----|----|----|----|----|----|----|----|
| Step mod 20 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| s | 5 | 11 | 7 | 13 | 15 | 6 | 13 | 9 | 5 | 11 |
| Step mod 20 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 |
| s | 7 | 12 | 8 | 15 | 13 | 8 | 15 | 6 | 7 | 14 |

Table 7. Constants

| Line | Round 1 | Round 2 | Round 3 | Round 4 | Round 5 |
|------|----------------------|----------------------------------|----------------------------------|----------------------------------|----------------------------------|
| X | 0 | 5a827999 [$2^{30}\sqrt{2}$] | 6ed9eba1 [$2^{30}\sqrt{3}$] | 8f1bbcdc [$2^{30}\sqrt{5}$] | a953fd4e [$2^{30}\sqrt{7}$] |
| Y | [$2^{30}\sqrt{7}$] | [$2^{30}\sqrt{5}$] | 0 | [$2^{30}\sqrt{2}$] | [$2^{30}\sqrt{3}$] |

Final Feedforward Process of Chaining Variables: Let us assume that the contents of the chaining variables before the compression function are $A \sim J$, and let the contents of the chaining variables after the compression function be $AA \sim JJ$. Then the updated contents of the chaining variables, or the output of the compression function, are given as shown.

$$A+ = AA, B+ = BB, C+ = CC, D+ = DD, E+ = EE, \\ F+ = FF, G+ = GG, H+ = HH, I+ = II, J+ = JJ.$$

Output Tailoring: In the case of 320-bit hash-code, the output is given as the contents of the 10 chaining variables concatenated, i.e. $A||B||C||D||E||F||G||H||I||J$. Otherwise, when the length of the hash-code is required to be shorter than 320 bits, it must be tailored into a string of specified length, denoted as $O_0||O_1||\dots||O_t$ ($t = 3, 4, \dots, 8$). The contents of O_i differ in each case of various lengths of hash-codes. Let us denote a t -bit string as $X^{[t]}$ to explicitly indicate the length of X .

- Case 1 (128-bit hash-code): The 32-bit chaining variables E and J can be divided as follows:

$$E = E_1^{[16]}E_0^{[16]}, \quad J = J_1^{[16]}J_0^{[16]}.$$

O_i is calculated as follows:

$$O_0 = A + F + E_1^{[16]}, \quad O_1 = B + G + E_0^{[16]}, \\ O_2 = C + H + J_1^{[16]}, \quad O_3 = D + I + J_0^{[16]}.$$

- Case 2 (160-bit hash-code): O_i is calculated as follow.

$$O_0 = A + F, \quad O_1 = B + G, \quad O_2 = C + H, \quad O_3 = D + I, \quad O_4 = E + J.$$

- Case 3 (192-bit hash-code): The 32-bit chaining variables D , E , I , and J can be divided as follows:

$$D = D_2^{[11]}D_1^{[11]}D_0^{[10]}, \quad E = E_2^{[11]}E_1^{[11]}E_0^{[10]}, \\ I = I_2^{[11]}I_1^{[11]}I_0^{[10]}, \quad J = J_2^{[11]}J_1^{[11]}J_0^{[10]}.$$

O_i is calculated as follows:

$$\begin{aligned} O_0 &= A + (J_2^{[11]} || I_1^{[11]}), & O_1 &= B + (J_1^{[11]} || I_0^{[10]}), \\ O_2 &= C + (J_0^{[10]} || I_2^{[11]}), & O_3 &= F + (E_2^{[11]} || D_1^{[11]}), \\ O_4 &= G + (E_1^{[11]} || D_0^{[10]}), & O_5 &= H + (E_0^{[10]} || D_2^{[11]}). \end{aligned}$$

- Case 4 (224-bit hash-code): The 32-bit chaining variables E , I , and J can be divided as follows:

$$E = E_2^{[11]} E_1^{[11]} E_0^{[10]}, \quad I = I_3^{[8]} I_2^{[8]} I_1^{[8]} I_0^{[8]}, \quad J = J_3^{[8]} J_2^{[8]} J_1^{[8]} J_0^{[8]}.$$

O_i is calculated as follows:

$$\begin{aligned} O_0 &= A + (J_3^{[8]} || I_2^{[8]}), & O_1 &= B + (J_2^{[8]} || I_1^{[8]}), \\ O_2 &= C + (J_1^{[8]} || I_0^{[8]}), & O_3 &= D + (J_0^{[8]} || I_3^{[8]}), \\ O_4 &= F + E_2^{[11]}, & O_5 &= G + E_1^{[11]}, & O_6 &= H + E_0^{[10]}. \end{aligned}$$

- Case 5 (256-bit hash-code): The 32-bit chaining variables E and J can be divided as follows:

$$E = E_3^{[8]} E_2^{[8]} E_1^{[8]} E_0^{[8]}, \quad J = J_3^{[8]} J_2^{[8]} J_1^{[8]} J_0^{[8]}.$$

O_i is calculated as follows:

$$\begin{aligned} O_0 &= A + J_3^{[8]}, & O_1 &= B + J_2^{[8]}, \\ O_2 &= C + J_1^{[8]}, & O_3 &= D + J_0^{[8]}, \\ O_4 &= F + E_3^{[8]}, & O_5 &= G + E_2^{[8]}, \\ O_6 &= H + E_1^{[8]}, & O_7 &= I + E_0^{[8]}. \end{aligned}$$

- Case 6 (288-bit hash-code): The 32-bit chaining variable J can be divided as follows:

$$J = J_4^{[7]} J_3^{[7]} J_2^{[6]} J_1^{[6]} J_0^{[6]}.$$

O_i is calculated as follow.

$$\begin{aligned} O_0 &= A + J_4^{[7]}, & O_1 &= B + J_3^{[7]}, \\ O_2 &= C + J_2^{[6]}, & O_3 &= D + J_1^{[6]}, & O_4 &= E + J_0^{[6]}, \\ O_5 &= F, & O_6 &= G, & O_7 &= H, & O_8 &= I. \end{aligned}$$

5 Design Rationales and Security Aspects

In this section, we discuss the underlying principles that were considered in the process of designing HAS-V. A great deal of attention was paid to balance the characteristics of security and performance. Security matters are considered based on previous attacks on hash functions and employ firm design philosophies of the previous hash functions. Performance matters are considered in the area of hardware support and algorithmic parallelism.

Number of Chaining Variables in Step Operation: One of the big differences between CISC processors, including the Intel 80x86 family and the Motorola 680x0, and RISC processors, including SPARC, MIPS, PA-RISC, PowerPC, and Alpha, is the number of on-chip general-purpose registers [4]. Generally RISC processors have enough general-purpose registers to load all the chaining variables on the register. However, due to their complex instruction set, CISC processors usually suffer from a shortage of general-purpose registers. In the case of a Pentium processor, there are only 7 general-purpose registers on its chip. Assuming at least 1 register is needed for temporary storage, no more than 6 registers can be used in an iterative step operation to perform without deterioration. HAVAL uses 8 chaining variables in its step operation and could suffer from performance deterioration in CISC processors. Therefore, in the design of HAS-V, a twin structure was employed and the number of chaining variables used in the step operation was chosen to be 5, so that the entire set of chaining variables could be loaded on the processor during the iterative process. It may seem at first that the two lines should be processed simultaneously since the chaining variables are swapped after every round. However, the two lines of the compression function can be processed independently as the entire chaining variables are swapped instead of just a portion of it. In an implementation point of view, the chaining variables are not actually swapped. Instead, the message words, Boolean function, and constants used in the step operation of round 2 and round 4 are replaced by the corresponding ones of the opposite line. This can be better understood by referring to the pseudo-code in Appendix A.

Process of Message Words: Early attacks on MD4 and MD5 were based on the weakness of the rather straightforward usage of the message words. An attack on the last two rounds of MD4 [1] and the cryptanalysis of MD4 [7] fall into this category of attack. A single message word of the input is only used once in every round of MD4 and MD5. This seems to provide vulnerability for inner collisions. A concept of message expansion was introduced in SHA-1, which provided a concrete security level against these sorts of attacks. This attractive property of message expansion was employed in the design of HAS-V. However, the generation of 64 message words from 16 message words of input seemed to load a heavy burden on the performance of SHA-1. In HAS-V, we have generated 20 message words from 16 message words for each line. This allows HAS-V to stay within a fairly good performance range, while providing enough diffusion from a single message word.

Step Operation: The structure of the step operation is an important factor in determining the performance factor. It is known that the step operation used in SHA-1 possesses a natural algorithmic parallelism in its compression function [4]. This arises from the fact that the updated chaining variable is not used in the Boolean function of the next step operation, which has the effect of reducing the critical path length. This mechanism has been employed in the step operation of HAS-V to provide a further advantage in performance. Other characteristics

in the step operation of HAS-V are the use of 4-variable Boolean functions and the use of a shift amount, which varies for each step within a single round of the compression function. The variable shift amount seems to provide better immunity against attacks such as differential collision in SHA-0 [5]. The generalization of inner collisions to a full compression function seemed to be harder with variable shift amounts.

Boolean Function: Previous attacks such as the collision attack on the compression function of MD5 [2] and differential attacks uses the linear approximation of Boolean functions. HAS-V employs 4-variable Boolean functions, whereas most MD4-family hash functions employ 3-variable Boolean functions. Having an extra variable in the Boolean function increases the complexity of a linear approximation and the computational cost of the Boolean function. Therefore, it is important to keep the balance between the needs for non-linearity and the loss of computational efficiency, while constructing a Boolean function. The computation of Boolean functions in HAS-V require about 3 or 4 unit operations¹, whereas the 3-variable Boolean functions used in other hash functions require about 2 or 3 unit operations. Therefore, by using 4-variable Boolean functions and omitting a single addition in the step function, we can improve the security aspects of HAS-V without performance deterioration. Among the numerous 4-variable Boolean functions, we have selected ones that are 0-1 balanced, satisfy SAC, and have a high non-linearity [16,17] to be used in HAS-V.

Output Tailoring: The output of HAS-V must provide a variable output from 128 bits to 320 bits, incrementing in multiples of 32 bits. We have modified the output tailoring method of HAVAL to produce the desired length of output, while providing a fair share to all of the chaining variables. Moreover, this process must not put unnecessary burden on the overall workload to deteriorate the performance.

Endianness: As with most of the MD4-family hash functions, the newly proposed hash function is optimized for 32-bit architecture processors. HAS-V favors 'little-endian' architectures. Processors with 'big-endian' architectures have to byte-reverse each word before processing, and since the big-endian processors are generally faster, it was decided to let them do the reversing it. This incurs a performance penalty of about 25%.

¹ The number of unit operations for Boolean functions can be defined by the least number of bit-wise operations such as \neg , \wedge , \vee , or \oplus , which are required to compute the Boolean functions. It can be regarded as a performance measure. If we modify the Boolean functions f_0 , f_1 , and f_2 , of HAS-V, the number of unit operations can be found. For example, since the truth table of f_0 is equivalent to that of $(x \oplus u)(y \oplus z) \oplus z$, the number of unit operations of f_0 is less than 4. In a similar way, those of f_1 and f_2 are 3 and 4, respectively

Table 8. Comparison of speed performance

| Algorithm | Performance(Mbits/sec) | |
|---------------|------------------------|------------------------|
| | Pentium III (600MHz) | Ultra 2 SPARC (300MHz) |
| | Microsoft Visual C++ | GNU C |
| MD5 | 41.95 | 22.04 |
| SHA-1 | 22.58 | 10.54 |
| RIPEMD-160 | 19.38 | 8.94 |
| HAVAL(5 PASS) | 20.64 | 11.01 |
| HAS-V | 27.86 | 12.58 |

6 Performance Evaluation

In this section we compare the performance of MD5, SHA-1, RIPEMD-160, HAVAL, and HAS-V. Output tailoring was ignored in both HAVAL and HAS-V, since it only occupies a negligible amount of time. Implementations were written in the C language and there was no optimization done in any way. The implementation was done solely for comparative reasons. The performance results were extracted by hashing 64Mbytes of data using an 8Kbyte buffer. Table 8 shows the results of our experiment. The results show that HAS-V has better performance than SHA-1, RIPEMD-160, or HAVAL with 5 passes in both environments.

The step operation of HAS-V consists of 3 additions, 2 circular shifts, and a Boolean function. Since the Boolean function consists of 4 unit operations, a single step operation will consist of 9 unit operations, assuming both addition and circular shift to be equivalent to unit operation. The total number of unit operations for generating the extra messages is $2(\text{lines}) \times 5(\text{rounds}) \times 4(\text{messages}) \times 3(\text{unit operations}) = 120(\text{unit operations})$. Therefore the number of unit operations to hash 1024 bit block is given below.

$$\begin{aligned}
 &1(\text{block}) \times 200(\text{steps}) \times 9(\text{step operation}) + 120(\text{message expansion}) \\
 &= 1920(\text{unit operations})
 \end{aligned}$$

In the case of RIPEMD-160, the total number of unit operation to hash 1024 bit block is given below.

$$\begin{aligned}
 &2(\text{block}) \times 160(\text{steps}) \times 9(\text{step operation}) \\
 &= 2880(\text{unit operations})
 \end{aligned}$$

This is about 33% more operation than HAS-V. This fact can also be seen in Table 8 where HAS-V is 31% faster than RIPEMD-160 on a Pentium PC.

7 Conclusion

We have proposed a new hash function with a variable length hash-code, namely HAS-V. The design was made such that it is both secure and efficient in most

computing environments. We expect that our results will broaden the use of KCDSA or any other cryptographic application that uses hash functions. We believe that the variable nature of the hash-code length will anticipate the needs of various practical applications.

Acknowledgements

Authors wish to express thanks to an anonymous referee of FSE2000 for pointing out a weakness of HAS-V in the previous version. This research was supported mainly by KISA(Korea Information Security Agency), partially by Brain Korea 21 and Com²MaC(Combinatorial and Computational Mathematics Center, POSTECH).

A Pseudo-Code of HAS-V

A.1 Definition

Let the input message string be consisted of t 1024-bit blocks, represented as $\{X_i[j], Y_i[j]\} (0 \leq i < t, 0 \leq j < 16)$.

$$\begin{aligned}
 f_j(x, y, z, u) &= xy \oplus \neg xz \oplus yu \oplus zu & (0 \leq j \leq 19) \\
 f_j(x, y, z, u) &= xz \oplus y \oplus u & (20 \leq j \leq 39) \\
 f_j(x, y, z, u) &= xy \oplus \neg xu \oplus z & (40 \leq j \leq 59) \\
 f_j(x, y, z, u) &= x \oplus yz \oplus u & (60 \leq j \leq 79) \\
 f_j(x, y, z, u) &= \neg xy \oplus xz \oplus yu \oplus zu & (80 \leq j \leq 99) \\
 g_j(x, y, z, u) &= \neg xy \oplus xz \oplus yu \oplus zu & (0 \leq j \leq 19) \\
 g_j(x, y, z, u) &= x \oplus yz \oplus u & (20 \leq j \leq 39) \\
 g_j(x, y, z, u) &= xy \oplus \neg xu \oplus z & (40 \leq j \leq 59) \\
 g_j(x, y, z, u) &= xz \oplus y \oplus u & (60 \leq j \leq 79) \\
 g_j(x, y, z, u) &= xy \oplus \neg xz \oplus yu \oplus zu & (80 \leq j \leq 99)
 \end{aligned}$$

$$\begin{aligned}
 K_j &= 00000000 & K'_j &= a953fd4e & (0 \leq j \leq 19) \\
 K_j &= 5a827999 & K'_j &= 8f1bbcdc & (20 \leq j \leq 39) \\
 K_j &= 6ed9eba1 & K'_j &= 00000000 & (40 \leq j \leq 59) \\
 K_j &= 8f1bbcdc & K'_j &= 5a827999 & (60 \leq j \leq 79) \\
 K_j &= a953fd4e & K'_j &= 6ed9eba1 & (80 \leq j \leq 99)
 \end{aligned}$$

$$s(j) = 5, 11, 7, 13, 15, 6, 13, 9, 5, 11, 7, 12, 8, 15, 13, 8, 15, 6, 7, 14$$

$$\begin{aligned}
 m(j) &= 18, 0, 1, 2, 3, 19, 4, 5, 6, 7, 16, 8, 9, 10, 11, 17, 12, 13, 14, 15 & (0 \leq j \leq 19) \\
 m(j) &= 18, 3, 6, 9, 12, 19, 15, 2, 5, 8, 16, 11, 14, 1, 4, 17, 7, 10, 13, 0 & (20 \leq j \leq 39) \\
 m(j) &= 18, 12, 5, 14, 7, 19, 0, 9, 2, 11, 16, 4, 13, 6, 15, 17, 8, 1, 10, 3 & (40 \leq j \leq 59) \\
 m(j) &= 18, 7, 2, 13, 8, 19, 3, 14, 9, 4, 16, 15, 10, 5, 0, 17, 11, 6, 1, 12 & (60 \leq j \leq 79) \\
 m(j) &= 18, 15, 9, 5, 3, 19, 12, 8, 6, 2, 16, 13, 11, 7, 1, 17, 14, 10, 4, 0 & (80 \leq j \leq 99)
 \end{aligned}$$

$a_j(k) = 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15$ ($j = 0$)
 $a_j(k) = 3, 6, 9, 12, 15, 2, 5, 8, 11, 14, 1, 4, 7, 10, 13, 0$ ($j = 20$)
 $a_j(k) = 12, 5, 14, 7, 0, 9, 2, 11, 4, 13, 6, 15, 8, 1, 10, 3$ ($j = 40$)
 $a_j(k) = 7, 2, 13, 8, 3, 14, 9, 4, 15, 10, 5, 0, 11, 6, 1, 12$ ($j = 60$)
 $a_j(j) = 15, 9, 5, 3, 12, 8, 6, 2, 13, 11, 7, 1, 14, 10, 4, 0$ ($j = 80$)

A.2 Pseudo-Code

Initial values of the chaining variables

$h_0 = 67452301; h_1 = e f c d a b 89; h_2 = 98 b a d c f e; h_3 = 10325476; h_4 = c3d2e1f0;$
 $h_5 = 8796a5b4; h_6 = 4b5a6978; h_7 = 0f1e2d3c; h_8 = a0b1c2d3; h_9 = 68794e5f;$

```

for  $i = 0, \dots, t - 1$  {
   $A = h_0; B = h_1; C = h_2; D = h_3; E = h_4;$ 
   $F = h_5; G = h_6; H = h_7; I = h_8; J = h_9;$ 
  for  $j = 0$  to  $99$  {
    if ( $j = 0, 40, 80$ ) {
      for  $k = 0$  to  $3$  {
         $X_i[16 + k] = X_i[a_j(4k)] \oplus X_i[a_j(4k + 1)] \oplus X_i[a_j(4k + 2)]$ 
           $\oplus X_i[a_j(4k + 3)];$ 
      }
    }
    else if ( $j = 20, 60$ ) {
      for  $k = 0$  to  $3$  {
         $Y_i[16 + k] = Y_i[a_j(4k)] \oplus Y_i[a_j(4k + 1)] \oplus Y_i[a_j(4k + 2)]$ 
           $\oplus Y_i[a_j(4k + 3)];$ 
      }
    }
    if ( $round = 1, 3, 5$ ) {
       $T = A \ll^{s(j \% 20)} + f_j(B, C, D, E) + X_i[m(j)] + K_j;$ 
    }
    else if ( $round = 2, 4$ ) {
       $T = A \ll^{s(j \% 20)} + g_j(B, C, D, E) + Y_i[m(j)] + K'_j;$ 
    }
     $E = D; D = C; C = B \ll^{30}; B = A; A = T;$ 
  }
  for  $j = 0$  to  $99$  {
    if ( $j = 0, 40, 80$ ) {
      for  $k = 0$  to  $3$  {
         $Y_i[16 + k] = Y_i[a_j(4k)] \oplus Y_i[a_j(4k + 1)] \oplus Y_i[a_j(4k + 2)]$ 
           $\oplus Y_i[a_j(4k + 3)];$ 
      }
    }
    else if ( $j = 20, 60$ ) {
      for  $k = 0$  to  $3$  {
         $X_i[16 + k] = X_i[a_j(4k)] \oplus X_i[a_j(4k + 1)] \oplus X_i[a_j(4k + 2)]$ 

```

$$\begin{aligned}
& \oplus X_i[a_j(4k+3)]; \\
& \} \\
& \} \\
& \text{if}(\text{round} = 1, 3, 5)\{ \\
& \quad T = F^{\ll s(j\%20)} + g_j(G, H, I, J) + Y_i[m(j)] + K'_j; \\
& \} \\
& \text{else if}(\text{round} = 2, 4)\{ \\
& \quad T = F^{\ll s(j\%20)} + f_j(G, H, I, J) + X_i[m(j)] + K_j; \\
& \} \\
& \quad J = I; I = H; H = G^{\ll 30}; G = F; F = T; \\
& \} \\
& \quad h_0 += F; h_1 += G; h_2 += H; h_3 += I; h_4 += J; \\
& \quad h_5 += A; h_6 += B; h_7 += C; h_8 += D; h_9 += E; \\
& \}
\end{aligned}$$

B Test Values of HAS-V

The test values of HAS-V are given in the case of 320-bit hash-code with no output tailoring

```

HAS-V("")=475974be d7ea137d 982d1df5 b2583b1a c4d5941d
8d557bb3 03586742 d8891788 943a9668 a9da68c3
HAS-V("abc")=a70ab818 294865cf 9c9697d6 97152353 70381b83
3f8f1a42 e0150588 8b002e43 05fe6405 519f595c

```

References

1. B.den Boer and A.Bosselaers, An attack on the last two rounds of MD4, *Advances in Cryptology-Crypto'91*, LNCS 576, Springer-Verlag, 1992, pp.194-203.
2. B.den Boer and A.Bosselaers, Collisions for the compression function of MD5, *Advances in Cryptology-Eurocrypt'93*, LNCS 773, Springer-Verlag, 1994, pp.293-304.
3. A.Bosselaers, R.Govaerts and J.Vandewalle, Fast hashing on the Pentium, *Advances in Cryptology-Crypto'96*, LNCS 1109, Springer-Verlag, 1996, pp.298-312.
4. A.Bosselaers, R.Govaerts and J.Vandewalle, SHA: a design for parallel architecture, *Advances in Cryptology-Eurocrypt'97*, 1997, pp.348-362.
5. F.Chabaud and A.Joux, Differential collisions in SHA-0, *Advances in Cryptology-Crypto'98*, LNCS 1462, Springer-Verlag, 1998, pp.56-71.
6. I.B.Damgård, A design principle for hash functions, *Proceedings of Crypto '89*, LNCS 435, Springer-Verlag, 1990, pp. 416-427.
7. H.Dobbertin, Cryptanalysis of MD4, *Fast Software Encryption*, LNCS 1039, Springer-Verlag, 1996, pp.53-69.
8. H.Dobbertin, The status of MD5 after a recent attack, *CryptoBytes*, 2(2), Sep. 1996, pp.1-6.
9. H.Dobbertin, A.Bosselaers and B.Preneel, RIPEMD160: A strengthened version of RIPEMD, *Fast Software Encryption*, LNCS1039, Springer-Verlag, 1996, pp.71-82. (An updated and corrected version is available at [ftp.esat.kuleuven.ac.be, /pub/COSIC/bosselaers/ripemd/](ftp.esat.kuleuven.ac.be/pub/COSIC/bosselaers/ripemd/).)

10. C.H.Lim and P.J.Lee, A study on the proposed Korean digital signature algorithm, *Advances in Cryptology-Asiacrypt'98*, LNCS 1514, Springer-Verlag, 1998, pp. 175-186.
11. NIST, Secure hash standard, *FIPS PUB 180-1*, Department of Commerce, Washington D.C., Apr. 1995.
12. P.C.van Oorschot and M.J.Wiener, Parallel collision search with applications to hash functions and discrete logarithms. *Proc. of 2nd ACM Conference on Computer and Communications Security*, ACM Press, 1994, pp.210-218.
13. B.Preneel, Analysis and design of cryptographic hash functions, *PHD thesis*, Katholieke University Leuven, 1993.
14. R.Rivest, The MD4 message digest algorithm, *Advances in Cryptology-Crypto'90*, LNCS 537, Springer-Verlag, 1991, pp.303-311.
15. R.Rivest, The MD5 message digest algorithm, *RFC 1321*, Internet Activities Board, Internet Privacy Task Force, Apr. 1992.
16. J.Seberry and X.M.Zhang, Highly nonlinear 0-1 balanced boolean functions satisfying strict avalanche criterion, *Advances in Cryptology-Auscrypt'92*, LNCS 718, Springer-Verlag, 1993, pp.145-154.
17. J.Seberry, X.M.Zhang and Y.Zheng, Nonlinearity and propagation characteristics of balanced boolean functions, *Information and Computation*, 119, 1995, pp.1-13.
18. G.Yuval, How to swindle Rabin, *Cryptologia*, Vol. 3, No. 3, 1979, pp.187-189.
19. Y.Zheng, J.Pieprzyk and J.Seberry, HAVAL - A one-way hashing algorithm with variable length of output, *Advances in Cryptology-Auscrypt'92*, LNCS 718, Springer-Verlag, 1993, pp.83-104.

On Welch-Gong Transformation Sequence Generators

G. Gong and A.M. Youssef

Center for Applied Cryptographic Research,
Department of Combinatorics and Optimization,
University of Waterloo,
Waterloo, Ontario N2L 3G1, CANADA
{ggong,a2youssef}@cacr.math.uwaterloo.ca

Abstract. Welch-Gong (WG) transformation sequences are binary sequences of period $2^n - 1$ with 2-level auto correlation. These sequences were discovered by Golomb, Gong and Gaal in 1998 and verified for $5 \leq n \leq 20$. Later on, No, Chung and Yun found another way to construct the WG sequences and verified their result for $5 \leq n \leq 23$. Dillon first proved this result for odd n in 1998, and finally, Dobbertin and Dillon proved it for even n in 1999. In this paper, we investigate a two-faced property of the WG transformation sequences for application in stream ciphers and pseudo-random number generators. One is to present randomness or unpredictability of the WG transformation sequences. The other is to exhibit the security property of the WG transformations regarded as Boolean functions. It is shown that the WG transformation sequences, in addition to the known 2-level auto correlation, have three-level cross correlation with m-sequences, large linear span increasing exponentially with n and efficient implementation. Thus this is the first type of pseudo-random sequences with good correlation and statistic properties, large linear span and efficient implementation. When the WG transformation are regarded as Boolean functions, it is proved that they have high nonlinearity. A criterion for whether the WG transformations regarded as Boolean functions are r -resilient is derived. It is shown that the WG transformations regarded as Boolean functions have large linear span (this concept will be defined in this paper) and high degree.

Key words: Stream cipher, pseudo-random sequence (number) generator, auto/cross correlation, linear span, Boolean function, non-linearity, r -resilient property.

1 Introduction

Pseudo-random sequences have been widely used in communications and cryptography. In order to guarantee that the pseudo-random sequence generators have good randomness or unpredictability, we have the following criteria:

- Long period
- Balance property (Golomb Postulate 1 [5])

- Run property (Golomb Postulate 2)
- n -tuple distribution
- Two-level auto correlation (Golomb Postulate 3)
- Low-level cross correlation
- Large linear span and smooth increased linear span profiles

In the last three years, the study of binary sequences with 2-level autocorrelation has made significant progress. The researchers [4,6,12,14,16] have found a number of new classes of binary sequences with 2-level auto correlation. In general, a pseudo-random sequence generator which generates sequences with 2-level auto correlation can be resistant to a correlation attack. However it is not easy to design a pseudo-random sequence generator which can generate sequences having both 2-level auto correlation and large linear span and be efficient to implement as well. Fortunately, it happened that one of classes of new sequences with 2-level auto correlation, so-called the Welch-Gong transformation sequences, possesses all these three properties. On the other hand, this type of sequences has period $2^n - 1$. Any binary sequence of period $2^n - 1$ is related to a function from the finite field $GF(2^n)$ to the finite field $GF(2)$. Thus it is automatically related to a Boolean function in n variables. Thus there is a connection among binary sequences with period $2^n - 1$, polynomial functions from $GF(2^n)$ to $GF(2)$ and Boolean functions in n variables. Chang, Dai and Gong [1] tried to use this connection. I.e., they applied m -sequences with three-level cross correlation to construct Boolean functions with the maximal non-linearity. In [10], Gong and Golomb successfully tried again to utilize this connection. They applied tools in pseudo-random sequence design and analysis to analyze the S-boxes in DES (Data Encryption Standard). When they considered the relationship between sequences and functions, they realized that monomials, which correspond to m -sequences, are not secure when used as component functions in block ciphers. This leads to a concept of linear span for polynomial functions introduced in their recent work [10]. In this paper, we will investigate the Welch-Gong transformation sequences in a two-faced aspect. One is to present their randomness, i.e., auto correlation, cross correlation with m -sequences, the balance property, and linear span when we consider them as sequences. The other is to derive the nonlinearity, the resilient property, linear span and degree when they are regarded as Boolean functions.

This paper is organized as follows. In Section 2, we give the definition of Welch-Gong transformation sequences. In Section 3, we present the randomness properties of the Welch-Gong transformation sequences which include an irregular decimation property, statistic properties, cross correlation with m -sequences, the Hadamard transform, and the linear span. In Section 4, we derive the non-linearity and a criterion for the resilient property (Note. Since any Welch-Gong transformation is balanced, so the correlation immunity property becomes the resilient property). In Section 5, we discuss linear span for the Welch-Gong transformations regarded as Boolean functions and show their degrees. Section 6 is a conclusion. All proofs omitted from this extended abstract can be found in the full paper [7].

We conclude this section by providing some preliminaries for sequence designs. The reader is referred to [5] for shift register sequences, [11] for the theory of finite field, and [19] and [18] for the motivation and the original definitions of nonlinearity and the resilient property for Boolean functions.

We will use the following notation throughout the paper:

- $\mathbb{F}_q = GF(q)$, a finite field with q elements, and \mathbb{F}_q^* , the multiplication group of \mathbb{F}_q .
- $\mathbb{F}_2^n = \{\mathbf{x} = (x_0, x_1, \dots, x_{n-1}) | x_i \in \mathbb{F}_2\}$, a vector space over \mathbb{F}_2 of dimension n .
- $\mathbf{a} = \{a_i\}$, a sequence over \mathbb{F}_2 , i.e., $a_i \in \mathbb{F}_2$, is called a *binary sequence*. If \mathbf{a} is a periodic sequence with period v , then we also denote $\mathbf{a} = (a_0, a_1, \dots, a_{v-1})$, an element in \mathbb{F}_2^v .

A. Autocorrelation

If $\mathbf{a} = (a_0, a_1, \dots, a_{p-1})$ is a binary sequence with period p , its (periodic) autocorrelation function $C(\tau)$ is defined as

$$C(\tau) = \sum_{i=0}^{p-1} (-1)^{a_i + a_{i+\tau}}, \tau = 0, 1, \dots.$$

Here τ is a phase shift of the sequence $\{a_i\}$.

Ideal (2-Level) Autocorrelation: If

$$C(\tau) = \begin{cases} p & \text{if } \tau \equiv 0 \pmod{p}, \\ -1 & \text{otherwise,} \end{cases}$$

then we say that the sequence $\{a_i\}$ has the idea two-level autocorrelation function.

B. Cross Correlation

If $\mathbf{a} = (a_0, a_1, \dots, a_{p-1})$ and $\mathbf{b} = (b_0, b_1, \dots, b_{p-1})$ are two binary sequences with period p , their (periodic) cross correlation function $C_{\mathbf{a}, \mathbf{b}}(\tau)$ is defined as

$$C_{\mathbf{a}, \mathbf{b}}(\tau) = \sum_{i=0}^{p-1} (-1)^{a_i + \tau + b_i}, \tau = 0, 1, \dots.$$

Here τ is a phase shift of the sequence $\{b_i\}$.

C. Hamming Weight

Let $H(s) = |\{0 \leq i < 2^n - 1 | s_i = 1\}|$ if $s = \{s_i\}$ is a binary sequence with period $2^n - 1$ and $H(s) = |\{x \in \mathbb{F}_{2^n} | s(x) = 1\}|$ if $s = s(x)$ is a polynomial function from \mathbb{F}_{2^n} to \mathbb{F}_2 . In both cases, we call $H(s)$ the *Hamming weight* of s . For a positive integer $r = r_0 + r_1 2 + \dots + r_{n-1} 2^{n-1}$, $r_i \in \mathbb{F}_2$, $H(r) = |\{0 \leq i < n | r_i = 1\}|$ is also called the *Hamming weight* of the integer r .

2 Definition of Welch-Gong Transformation Sequence Generators

In this section, we will give the definition of the Welch-Gong transformation sequence generators. Hence after, we set $n \not\equiv 0 \pmod 3$.

Let $g(x) = x + x^{q_1} + x^{q_2} + x^{q_3} + x^{q_4}$, $x \in \mathbb{F}_{2^n}$, where q_i are defined by

$$\begin{aligned} q_1 &= 2^k + 1, \\ q_2 &= 2^{2k-1} + 2^{k-1} + 1, \\ q_3 &= 2^{2k-1} - 2^{k-1} + 1 \text{ and} \\ q_4 &= 2^{2k-1} + 2^k - 1, \end{aligned} \tag{1}$$

where $n = 3k - 1$ and

$$\begin{aligned} q_1 &= 2^{k-1} + 1, \\ q_2 &= 2^{2k-2} + 2^{k-1} + 1, \\ q_3 &= 2^{2k-2} - 2^{k-1} + 1 \text{ and} \\ q_4 &= 2^{2k-1} - 2^{k-1} + 1, \end{aligned} \tag{2}$$

where $n = 3k - 2$. Then a function, say $f(x)$, from \mathbb{F}_{2^n} to \mathbb{F}_2 defined by

$$f(x) = \text{Tr}(g(x + 1) + 1), x \in \mathbb{F}_{2^n} \tag{3}$$

is called the *Welch-Gong transformation* of $\text{Tr}(g(x))$, or the *WG transformation* for short.

Let α be a primitive element of \mathbb{F}_{2^n} . Let $\mathbf{a} = \{a_i\}$ and $\mathbf{b} = \{b_i\}$ whose elements are given by

$$a_i = \text{Tr}(g(\alpha^i)), b_i = f(\alpha^i) = \text{Tr}(g(\alpha^i + 1) + 1), i = 0, 1, \dots \tag{4}$$

Then \mathbf{b} is called a *Welch-Gong transformation sequence* of \mathbf{a} , or *WG sequence* for short.

Any function from \mathbb{F}_{2^n} to \mathbb{F}_2 is related to a Boolean function (we will discuss an exact conversion of these two representations in Section 4). From a WG transformation, we have two types of pseudo-random sequence generators. One is WG sequences themselves. The other is to apply WG transformations regarded as Boolean functions to operate on a set of LFSRs for generating sequences. I.e., applying the WG transformations regarded as Boolean functions as combining functions or filtering functions in combinatorial function generators or filtering generators [17]. We refer to these two modes as *WG sequence generators*.

Remark 1. In fact, the transform given in (3) can be applied to any function from \mathbb{F}_{2^n} to \mathbb{F}_2 . But till now, we haven't found another type of $g(x)$ such that its WG transformation sequence has 2-level auto correlation. So, we restrict ourselves to this specific $g(x)$.

3 Randomness of WG Sequences

In this section, we will discuss the randomness properties of WG sequences, including their decimation, auto/cross correlation, statistic properties, and linear span.

A. Decimation Property

Lemma 1. *Let α be a primitive element of \mathbb{F}_{2^n} and $\underline{\mathbf{b}}$ be the WG sequence of $\underline{\mathbf{a}}$. Then elements of $\underline{\mathbf{b}}$ can be obtained by operating an irregular decimation on $\underline{\mathbf{a}}$ as follows: $b_0 = a_0$ for $i > 0$,*

$$b_i = \begin{cases} a_{\tau(i)} & \text{if } n \text{ even,} \\ a_{\tau(i)} + 1 & \text{if } n \text{ odd,} \end{cases}$$

where $\tau(i)$ is determined by

$$\alpha^{\tau(i)} = \alpha^i + 1. \quad (5)$$

Proof. From the definition, $a_0 = \text{Tr}(g(1)) = n$ and $b_0 = \text{Tr}(g(0) + 1) = n$. Thus $a_0 = b_0$. For $i > 0$, if n is even, we have

$$\begin{aligned} b_i &= \text{Tr}(g(\alpha^i + 1) + 1) = \text{Tr}(g(\alpha^i + 1)) + n \\ &= \text{Tr}(g(\alpha^i + 1)) = a_{\tau(i)}, i = 0, 1, \dots \end{aligned} \quad (6)$$

Similarly, if n is odd, we have

$$\begin{aligned} b_i &= \text{Tr}(g(\alpha^i + 1) + 1) = \text{Tr}(g(\alpha^i + 1)) + n \\ &= \text{Tr}(g(\alpha^i + 1)) = a_{\tau(i)} + 1, i = 0, 1, \dots \end{aligned} \quad (7)$$

Thus the assertion is established.

Remark 2. This lemma shows that the WG sequence $\underline{\mathbf{b}}$ can be obtained by an irregular decimation from $\underline{\mathbf{a}}$ where the decimation is determined by (5). Note that $\underline{\mathbf{a}}$ is a 5-term sequence and it can be generated by using five linear feedback shift registers and one AND gate. This property of the WG sequences allows them to have an efficient implementation for small n by operating decimation on $\underline{\mathbf{a}}$ together with a table look-up.

B. Auto Correlation, 2-Tuple Distribution, and Balance Property

Proposition 1. *Let $\underline{\mathbf{b}}$ be a WG sequence defined by (4). Then $\underline{\mathbf{b}}$ is a binary sequence of period $2^n - 1$ with (ideal) 2-level auto correlation.*

The proof of this proposition can be found in [4].

Remark 3. This result was first discovered by Golomb, Gong and Gaal in [12] and verified for $5 \leq n \leq 20$. Later on, No *et al* [16], found another way to construct the WG sequences and verified their result for $5 \leq n \leq 23$. Dillon proved it for the odd n case [2], and finally, Dobbartin and Dillon proved it for the even n case [4] which completely established Proposition 1.

Note. $\underline{\mathbf{a}}$ is also a 2-level auto correlation sequence. See [12,4].

Let $\underline{\mathbf{s}}$ be a binary sequence of period $2^n - 1$. Assume that $1 \leq t \leq n$. In every period of $\underline{\mathbf{s}}$, if each nonzero t -tuple $(c_1, c_2, \dots, c_t) \in F_2^t$ occurs 2^{n-t} times and zero t -tuple $(0, \dots, 0)$ occurs $2^{n-t} - 1$ times, then we say that the sequence has an *ideal t -tuple distribution*.

Proposition 2. *Any WG sequence has ideal 2-tuple distribution and it is balanced (i.e., in every period $2^n - 1$, zeros occur $2^{n-1} - 1$ times and ones occur 2^{n-1} times).*

Proof. Let $\underline{\mathbf{b}}$ be a WG sequence of period $2^n - 1$. Since $\underline{\mathbf{b}}$ is a 2-level auto correlation sequence, then for any shift of $\underline{\mathbf{b}}$, say $L^\tau(\underline{\mathbf{b}}) = (b_\tau, b_{\tau+1}, \dots)$, we have

$$|\{0 \leq k < 2^n - 1 \mid (b_k, b_{k+\tau}) = (i, j)\}| = \begin{cases} 2^{n-2} - 1 & \text{if } i = j = 0, \\ 2^{n-2} & \text{otherwise,} \end{cases} \quad (8)$$

where $i, j \in \mathbb{F}_2$. When $\tau = 1$ it establishes the first assertion. The second result follows from the first one. (For a detail proof, please see the full paper.)

C. Hadamard Transform and Cross Correlation

Let $f(x)$ be a function from \mathbb{F}_{2^n} to \mathbb{F}_2 . Then the Hadamard transform of $f(x)$ is defined by

$$\hat{f}(\lambda) = \sum_{x \in \mathbb{F}_{2^n}} (-1)^{\text{Tr}(\lambda x) + f(x)}. \quad (9)$$

Property 1. Let $S_v(x) = \text{Tr}(x^v)$. Let $n = 2m + 1$ be odd and $v = 2^t + 1$ with $\gcd(t, n) = 1$. Then the Hadamard transform of $S_{2^t+1}(x)$ is given by

$$\hat{S}_{2^t+1}(\lambda) = \begin{cases} 0 & \text{if } \text{Tr}(\lambda) = 0, \\ \pm 2^{m+1} & \text{if } \text{Tr}(\lambda) = 1. \end{cases}$$

This result is established by Gold [8] in 1968.

Theorem 1. *For odd n , let $g(x)$ be defined as in Section 2.1, and let $f(x)$ be the Welch-Gong transform of $\text{Tr}(g(x))$, i.e., $f(x) = \text{Tr}(g(x+1) + 1)$. Then $\hat{f}(\lambda)$, the Hadamard transform of $f(x)$, is given by*

$$\hat{f}(\lambda) = \hat{S}_{2^t+1}(\lambda^c), \quad (10)$$

where

$$c = d^{-1} \text{ and } d = 2^{2t} - 2^t + 1 \text{ for } 3t \equiv 1 \pmod{n}. \quad (11)$$

Proof. Here we only give a link with the source for our proof. The linear span of WG sequences are give in [12] by Golomb, Gong and Gaal. Later on, No, Chung and Yin found another way to generate the WG sequences in [16] which

is verified by experimental results. A few months later, Dobbertin proved that the sequences generated in [16] are WG sequences in [3] through these two types of sequences having the same linear span. Dillon proved that the sequences generated by No-Chung-Yin have 2-level autocorrelation for n odd by establishing the Hadamard transform of an No-Chung-Yin sequence is equal to the value of the right hand of equation (10). Thus the result follows from the above link.

From Theorem 1 and Property 1, we now have the following corollaries.

Corollary 1 (Function Version). *Let $n = 2m + 1$ odd, and let $f(x)$ be the Welch-Gong transformation function. Then the Hadamard transform of $f(x)$ is given by*

$$\hat{f}(\lambda) = \begin{cases} 0 & \text{if } Tr(\lambda^c) = 0, \\ \pm 2^{m+1} & \text{if } Tr(\lambda^c) = 1, \end{cases}$$

where c is defined in (11) in Theorem 1.

Proof. Applying Property 1 to Theorem 1, the result follows.

Corollary 2 (Sequence Version). *Let α be a primitive element of \mathbb{F}_{2^n} and $f(x)$ be a Welch-Gong transformation function. Let $\underline{\mathbf{a}} = \{a_i\}$ be an m -sequence whose elements are defined by*

$$a_i = Tr(\alpha^i), i = 0, 1, \dots$$

and $\underline{\mathbf{b}} = \{b_i\}$ be the WG sequence whose elements are given by (4). Then $C_{\underline{\mathbf{a}}, \underline{\mathbf{b}}}(\tau)$, the cross correlation function between $\underline{\mathbf{a}}$ and $\underline{\mathbf{b}}$, is determined by

$$C_{\underline{\mathbf{a}}, \underline{\mathbf{b}}}(\tau) = \begin{cases} -1 & \text{if } Tr(\alpha^{\tau c}) = 0, \\ -1 \pm 2^{m+1} & \text{if } Tr(\alpha^{\tau c}) = 1, \end{cases}$$

where c is defined in (11) in Theorem 1. I.e., the cross correlation function between $\underline{\mathbf{a}}$ and $\underline{\mathbf{b}}$ are three-valued.

D. Linear Span of WG Sequences

The linear span of a sequence is defined to be the *shortest length* of the linear feedback shift registers which generate the sequence. Sequence with large linear span are resistant to attacks arising from employing the Berlekamp-Massey algorithm [15].

Proposition 3. *Let $\underline{\mathbf{b}}$ be a WG sequence of period $2^n - 1$ and $LS(\underline{\mathbf{b}})$ represent its linear span. Then*

$$LS(\underline{\mathbf{b}}) = n(2^{\lceil n/3 \rceil} - 3).$$

A proof for this result is given in Section 5. From Proposition 3, it is clear that the linear span of the WG sequences of period $2^n - 1$ increases exponentially with n .

Remark 4. WG sequences of period $2^n - 1$ are the first type of binary sequences of period $2^n - 1$ which have the balance property, ideal 2-tuple distribution, 2-level auto correlation, three-level cross correlation with m -sequences, linear span increased exponentially in n .

4 Non-linearity and Resilient Property of Welch-Gong Transformations

In this section, we will derive the non-linearity and the resilient property for the WG transformations when they are regarded as Boolean functions. First, we need to develop a result on the conversion of a polynomial function (from \mathbb{F}_{2^n} to \mathbb{F}_2) to a Boolean function in n variables.

4.1 Isomorphism between \mathbb{F}_{2^n} and \mathbb{F}_2^n

Since the finite field \mathbb{F}_{2^n} can be regarded as a vector space of n dimension, then we have a linear space structure for \mathbb{F}_{2^n} . Let $B = \{\alpha_0, \alpha_1, \dots, \alpha_{n-1}\}$ be a basis of \mathbb{F}_{2^n} over \mathbb{F}_2 , then $\forall x \in \mathbb{F}_{2^n}$, we have

$$x = x_0\alpha_0 + x_1\alpha_1 + \dots + x_{n-1}\alpha_{n-1}, x_i \in \mathbb{F}_2.$$

Let

$$\delta : x \mapsto \underline{x} = (x_0, x_1, \dots, x_{n-1}) \quad (12)$$

then $\delta(x)$ is an isomorphism between \mathbb{F}_{2^n} and \mathbb{F}_2^n when both of them are regarded as vector spaces.

Let $f(x)$ be a function from \mathbb{F}_{2^n} to \mathbb{F}_2 . Then $f(x)$ defines a Boolean function in the following way:

$$f(x) = f(x_0\alpha_0 + \dots + x_{n-1}\alpha_{n-1}) = f_B(x_0, x_1, \dots, x_{n-1})$$

Note that f and f_B in the above identity might not be same. We will write $f_B(x_0, x_1, \dots, x_{n-1}) = f(x_0, x_1, \dots, x_{n-1})$ for short if it will not cause any confusion in the context.

Remark 5. For a given Boolean function in n variables, we can obtain its polynomial representation which is a function from \mathbb{F}_{2^n} to \mathbb{F}_2 by using so-called the Fourier transform, see [10].

4.2 Non-linearity of WG Transformations

Let $f(\underline{x})$, $\underline{x} = (x_0, x_1, \dots, x_{n-1}) \in \mathbb{F}_2^n$, be a Boolean function. A Boolean function $a(\underline{x})$ is said to be *affine* if

$$a(\underline{x}) = \sum_{i=0}^{n-1} w_i x_i + c, w_i \in \mathbb{F}_2, c \in \mathbb{F}_2.$$

Let

$$A = \left\{ \sum_i w_i x_i + c \mid w_i \in \mathbb{F}_2, c \in \mathbb{F}_2 \right\}.$$

I.e., A is the set consisting of all affine Boolean functions in n variables. Then the non-linearity of $f(\underline{\mathbf{x}})$, denoted by N_f , is defined as

$$N_f = \min_{a \in A} d(f, a)$$

where $d(f, a) = |\{\underline{\mathbf{x}} \in F_2^n | f(\underline{\mathbf{x}}) \neq a(\underline{\mathbf{x}})\}|$ which is called a *distance* of f from a .

Remark 6. From the definition of the distance, we have $d(f, a) = H(f + a)$, where $H(f + a)$ is the Hamming weight of $f + a$ defined in Section 1.

Note that

$$d(f, a) = |\{x \in \mathbb{F}_{2^n} | f(x) \neq a(x)\}|$$

where $f(x)$ and $a(x)$ are polynomial forms of the Boolean functions $f(\underline{\mathbf{x}})$ and $a(\underline{\mathbf{x}})$ respectively. I.e., the distance of f from a is not changed whenever either a Boolean form or a polynomial form are applied.

Let

$$\hat{f}(\underline{\mathbf{w}}) = \sum_{\underline{\mathbf{x}} \in \mathbb{F}_2^n} (-1)^{\underline{\mathbf{w}} \cdot \underline{\mathbf{x}} + f(\underline{\mathbf{x}})} \quad (13)$$

which is called the *Walsh transform* of the Boolean function f in the literature.

Theorem 2. Let $n = 2m + 1$ odd. Let $f(x)$ be the Welch-Gong transformation defined by (3). Let $f(\underline{\mathbf{x}})$ be the Boolean form of $f(x)$. Then the non-linearity of $f(\underline{\mathbf{x}})$, denoted by N_f , is given by

$$N_f = 2^{n-1} - 2^m.$$

In order to prove Theorem 2, we need the following two lemmas whose proofs can be found in the full paper.

Lemma 2. Let $f(\underline{\mathbf{x}})$ be a Boolean function and $a(\underline{\mathbf{x}})$ be a linear Boolean function. Let $f(x)$ and $a(x)$ be a polynomial representation of $f(\underline{\mathbf{x}})$ and $a(\underline{\mathbf{x}})$ respectively.

1. $a(\underline{\mathbf{x}}) = \sum_i w_i x_i = \underline{\mathbf{w}} \cdot \underline{\mathbf{x}}$ where $\underline{\mathbf{w}} = (w_0, w_1, \dots, w_{n-1}) \in \mathbb{F}_2^n$. Moreover, there exists some $\lambda \in \mathbb{F}_{2^n}$ such that $a(\underline{\mathbf{x}}) = \text{Tr}(\lambda x)$.
2. The Hadamard transform of $f(x)$ and the Walsh transform of $f(\underline{\mathbf{x}})$ have the following relation:

$$\hat{f}(\underline{\mathbf{w}}) = \hat{f}(\lambda), \underline{\mathbf{w}} \in \mathbb{F}_2^n, \lambda \in \mathbb{F}_{2^n} \quad (14)$$

where $\underline{\mathbf{w}} \cdot \underline{\mathbf{x}} = \text{Tr}(\lambda x)$.

3. The Hadamard transform of f and the distance of $f(\underline{\mathbf{x}})$ from $a(\underline{\mathbf{x}})$ are related by

$$\hat{f}(\lambda) = 2^n - 2d(f, a)$$

where λ is the same as above. Or equivalently,

$$d(f, a) = \frac{2^n - \hat{f}(\lambda)}{2}.$$

Lemma 3. *Let $f(\underline{x})$ be a function from \mathbb{F}_2^n to \mathbb{F}_2 . Let $a(\underline{x})$ be a linear function from \mathbb{F}_2^n to \mathbb{F}_2 . Then $d(f, a+1) = 2^n - d(f, a)$. Moreover $d(f, a+1) = \frac{2^n + \hat{f}(\lambda)}{2}$ where λ satisfies that $a(\underline{x}) = \text{Tr}(\lambda x)$.*

Proof of Theorem 2. Applying Lemma 2-(3), Corollary 1, and Lemma 3, the assertion follows. (A detailed proof is included in the full paper.)

Remark 7. Chang, Dai and Gong [1] discussed how to construct Boolean functions with the maximal non-linearity in terms of binary m-sequences with three-valued cross correlation. In particular, they discussed the case $\text{Tr}(x^r)$ for some special choices of r . Gong and Golomb [10], pointed out that the monomial functions $\text{Tr}(x^r)$ are not secure when used as combining functions or filtering functions in stream cipher systems or block cipher modes, because they correspond to m-sequences. However, the WG transformations are not monomials. We will come back to this question in the next section.

4.3 The Resilient Property of WG Transformations

Let $f(\underline{x}), \underline{x} \in \mathbb{F}_2^n$, be a Boolean function. For $r > 0$, $f(\underline{x})$ is said to be *r-order correlation immune* if

$$\hat{f}(\underline{w}) = 0 \text{ for all } \underline{w} \in \mathbb{F}_2^n : 1 \leq H(\underline{w}) \leq r. \quad (15)$$

This definition comes from the result obtained by Xiao and Massey [20] which is equivalent to Siegenthaler's original definition [18]. If $f(\underline{x})$ satisfies (15) and $f(\underline{x})$ is balanced, i.e., $H(f(\underline{x})) = 2^{n-1}$. Then $f(\underline{x})$ is said to be *r-resilient*.

Let

$$D = \{x \in \mathbb{F}_{2^n}^* | \text{Tr}(x^c) = 0\} \quad (16)$$

where c is defined in Theorem 1. Then $|D| = 2^{n-1} - 1$. Recall that $\{\alpha_0, \alpha_1, \dots, \alpha_{n-1}\}$ is the basis of \mathbb{F}_{2^n} over \mathbb{F}_2 . Let

$$R = \{(\text{Tr}(\lambda\alpha_0), \dots, \text{Tr}(\lambda\alpha_{n-1})) \in \mathbb{F}_2^n | \lambda \in D\}. \quad (17)$$

Theorem 3. *With the above notation. Let n be odd. Let $f(\underline{x})$ be the Boolean form of the WG transformation $f(x) = \text{Tr}(g(x+1)+1)$ defined by (3). Then $f(\underline{x})$ is *r-resilient* if and only if all vectors in $W_r = \{\underline{w} | 1 \leq H(\underline{w}) \leq r\}$ appear in R . I.e., $W_r \subset R$.*

Proof. Applying Lemma 2, we have $\hat{f}(\underline{w}) = \hat{f}(\lambda)$ where

$$a(\underline{x}) = \sum_{i=0}^{n-1} w_i x_i = \text{Tr}(\lambda x). \quad (18)$$

Notice that

$$\lambda x = \lambda \sum_i x_i \alpha_i \implies \text{Tr}(\lambda x) = \sum_{i=0}^{n-1} \text{Tr}(\lambda \alpha_i) x_i.$$

Combining with (18), we have $w_i = Tr(\lambda\alpha_i), 0 \leq i < n$. According to Corollary 1, $\hat{f}(\lambda) = 0$ if and only if $Tr(\lambda^c) = 0$ where c is defined by (11). Together with the definition of the resilient property, the assertion follows.

Note. Theorem 3 provides a method to find r -resilient WG transformations regarded as Boolean functions.

In the following we will derive that any WG-transformation regarded as a Boolean function with 1-resilient property is always possible by a proper basis conversion.

Theorem 4. *Let $f(x)$ be a WG-transformation. Then there exists at least one basis of \mathbb{F}_{2^n} such that the Boolean function representation of $f(x)$ under this basis is 1-resilient*

Proof. First we state the following claim whose proof can be found in the full paper.

Claim. There are n linearly independent vectors in R , defined by (17).

Therefore we can assume that $\{\alpha_0, \dots, \alpha_{n-1}\}$ is a subset of $R_{\underline{\alpha}}$ which is linearly independent over \mathbb{F}_2 . Let A be an $n \times n$ matrix with row vectors $\alpha_i, i = 0, \dots, n-1$. Let $B = A^{-1}$ and let β_j denote the j th column vector of $B, j = 0, \dots, n-1$. Then $\underline{\beta} = \{\beta_0, \dots, \beta_{n-1}\}$ is a basis of \mathbb{F}_{2^n} . Then $R_{\underline{\beta}}$ is given by

$$R_{\underline{\beta}} = \{\sigma_{\underline{\alpha}}(\lambda)B | \lambda \in D\}.$$

Therefore the row vectors of $BA = A^{-1}A = I_n$, the identity matrix, belong to $R_{\underline{\beta}}$. Hence $f_{\underline{\beta}}(x_0, \dots, x_{n-1})$, the Boolean representation of $f(x)$ under the basis $\underline{\beta}$, is a 1-resilient function.

5 Linear Span and Degree of WG Transformations

Let $f(x) = \sum_i c_i x^i$ be a polynomial function from \mathbb{F}_{2^n} to \mathbb{F}_2 . Let $I = \{0 \leq i \leq 2^n - 1 | c_i \neq 0\}$. The *algebraic degree* of $f(x)$, denoted as $alg(f)$, is defined as

$$alg(f) = \max_{i \in I} alg(x_i) \text{ where } alg(x_i) = H(i).$$

Let $f(\underline{x})$ be the Boolean form of $f(x)$ and denote the degree of $f(\underline{x})$ as $deg(f(\underline{x}))$.

Fact 1 *The algebraic degree of $f(x)$, a polynomial function from \mathbb{F}_{2^n} to \mathbb{F}_2 , is equal to the degree of the Boolean form of the function. I.e., $alg(f) = deg(f(\underline{x}))$.*

Linear span of $f(x)$ is said to be the number of non-zero coefficients in $f(x) = \sum_i c_i x^i$, which is introduced by Gong and Golomb in [10]. We denote it as $LS(f(x))$ or simply $LS(f)$ if the context is clear. I.e., $LS(f) = |I|$.

Note. The linear span of a polynomial function from \mathbb{F}_{2^n} to \mathbb{F}_2 is equal to the linear span of the sequence corresponding to the function.

Let $f(\underline{x})$ be a Boolean function, we will define the linear span of the Boolean function $f(\underline{x})$ in terms of its polynomial representations. Let

$$\Pi = \{ \text{all bases of } \mathbb{F}_{2^n} \text{ over } \mathbb{F}_2 \}.$$

For $B = \{\alpha_0, \dots, \alpha_{n-1}\} \in \Pi$, we denote $f_B(x)$ a polynomial representation of $f(\underline{x})$ with respect to the basis B . We define a *linear span* of $f(\underline{x})$, denoted by $LS(f(\underline{x}))$, as

$$LS(f(\underline{x})) = \min_{B \in \Pi} LS(f_B(x)).$$

Note that for a given polynomial function $f(x)$, the linear span of any Boolean representation of $f(x)$ is equal to the linear span of $f(x)$ itself. We will write this observation as a lemma for later reference.

Lemma 4. *With the above notation. Let $f(x)$ be a polynomial function of \mathbb{F}_{2^n} to \mathbb{F}_2 . Let $f_B(\underline{x})$ be its Boolean form with respect to a basis B of \mathbb{F}_{2^n} over \mathbb{F}_2 . Then $LS(f_B(\underline{x})) = LS(f(x))$ for all $B \in \Pi$.*

As Youssef and Gong pointed it out in their recent work [21], the polynomial representation a complicated Boolean function might be just a monomial function (here monomial means that it has only one trace term in (19) which is different from the concept of the ordinary monomial which only has exactly one term). Thus a Boolean function must have a large linear span so that it can be resistant to the interpolation attack. In the following, we will show that the linear span of the Boolean forms of the WG transformations increases exponentially with n .

We have the following result whose proof can be found in [12].

Proposition 4. *Let $f(x) = \text{Tr}(g(x+1)+1)$ be the WG transformation defined by (3), then*

$$f(x) = \sum_{i \in I} \text{Tr}(x^i) \quad (19)$$

where $I = I_1 \cup I_2$ for $n = 3k - 1$, where

$$\begin{aligned} I_1 &= \{2^{2k-1} + 2^{k-1} + 2 + i \mid 0 \leq i \leq 2^{k-1} - 3\}, \text{ and} \\ I_2 &= \{2^{2k} + 3 + 2i \mid 0 \leq i \leq 2^{k-1} - 2\} \end{aligned} \quad (20)$$

and where $I = \{1\} \cup I_3 \cup I_4$ for $n = 3k - 2$, where

$$\begin{aligned} I_3 &= \{2^{k-1} + 2 + i \mid 0 \leq i \leq 2^{k-1} - 3\}, \text{ and} \\ I_4 &= \{2^{2k-1} + 2^{k-1} + 2 + i \mid 0 \leq i \leq 2^{k-1} - 3\}. \end{aligned} \quad (21)$$

Moreover, in each case, all the elements in I belong to distinct cyclotomic cosets modulo $2^n - 1$.

Note that the trace functions appeared in (19) depend on the coset size of i in I .

Theorem 5. Let $f(x) = \text{Tr}(g(x+1) + 1)$ be the WG transformation defined by (3), then $LS(f(x))$, the linear span of f , is given by $LS(f(x)) = n(2^{\lceil n/3 \rceil} - 3)$.

Proof. According to Fact 4, all numbers in I belong to different cyclotomic cosets modulo $2^n - 1$ and $|I| = 2^k - 3$. So we only need to show that any coset containing a number in I has full size n . Here we only give a proof for $n = 3k - 1$ and $s \in I_1$, since proofs for the other cases are similar. Note that for $s \in I_1$, the binary representation of s has the following pattern

$$\begin{array}{ll} \text{Index} & 0 \ 1 \cdots k-1 \ k \cdots 2k-2 \ 2k-1 \ 2k \cdots 3k-2 = n-1 \\ \text{BinaryRep.} & * \ * \cdots \ * \ 0 \cdots \ 0 \quad 1 \quad 0 \cdots \ 0 \end{array} \quad (22)$$

where $*$ can take any value from $\{0, 1\}$. Let C_s be a coset containing s , then

$$C_s = \{s, s2, \dots, s2^{n_s-1}\}$$

where n_s is the smallest integer satisfying $s2^{n_s} \equiv s \pmod{2^n - 1}$. According to (22), $n_s = n$. I.e., C_s has full size n . Thus for each $s \in I$, we have the trace function appeared in Fact 4 is the trace function from \mathbb{F}_{2^n} to \mathbb{F}_2 . Thus $LS(\text{Tr}(x^i)) = n$ for each $i \in I$. Therefore the result follows.

Proof of Proposition 3 in Section 3. Since the linear span of a WG sequence \mathbf{b} is equal to the linear span of the corresponding WG transformation f . I.e., we have $LS(\mathbf{b}) = LS(f)$. Applying Theorem 5, the result follows.

Theorem 6. Let $f(x)$ be the WG transformation defined by (3). Then the linear span of any Boolean form of $f(x)$ is equal to the linear span of $f(x)$. I.e., $LS(f(\mathbf{x})) = n(2^{\lceil n/3 \rceil} - 3)$.

Proof. The result follows from Lemma 4 and Theorem 5.

Theorem 7. Let $f(x) = \text{Tr}(g(x+1) + 1)$ be the WG transformation defined by (3) and $f(\mathbf{x})$ be its Boolean form. Then $\deg(f(\mathbf{x}))$, the degree of $f(\mathbf{x})$, is given by $\deg(f(\mathbf{x})) = \lceil n/3 \rceil + 1$.

Proof. According to Fact 1, the degree of $f(\mathbf{x})$ is equal to the algebraic degree of $f(x)$. From the definition, the algebraic degree of $f(x)$ is determined by the largest Hamming weight among the integers appeared in the set I in Fact 4. We can easily verify that $k+1$ for both $n = 3k - 1$ and $n = 3k - 2$ is such number. Thus the assertion is established. (See the full paper, for a detailed proof.)

Applying the Siegenthaler inequality [18] and Theorem 7, the following corollary is immediate.

Corollary 3. For a given WG transformation regarded as a Boolean function in n variables, r , the order of the resilient property of the function, is bounded by the following inequality $r \leq n - \lceil n/3 \rceil$.

6 Example

In this section, we will give an example to illustrate the randomness properties of WG transformations regarded as both WG sequences and Boolean functions, which we obtained in the previous sections. Let \mathbb{F}_{2^7} be defined by a primitive polynomial $h(x) = x^7 + x + 1$. Let α be a root of $h(x)$. Then α is a primitive element of \mathbb{F}_{2^7} . Since $n = 7$, then $k = 3$, and

$$q_1 = 5, q_2 = 21, q_3 = 13, \text{ and } q_4 = 29 \text{ and } g(x) = x + x^5 + x^{21} + x^{13} + x^{29}$$

So, the WG transformation is

$$f(x) = \text{Tr}(g(x+1) + 1) = \text{Tr}(x + x^3 + x^7 + x^{19} + x^{29}).$$

A. Sequence Aspects of the WG transformation:

We obtain a WG sequence $\underline{b} = \{b_i\}$ as follows:

$$\underline{b} = 1000000101000011011100010100101100101010000101100010010111011011000110011101100100000110001111010101110100100111111100111101111$$

where $b_i = f(\alpha^i), i = 0, 1, \dots$. The WG sequence \underline{b} has the Balance property, ideal 2-tuple distribution, 2-level auto correlation, 3-valued cross correlation with an m-sequence defined by $\{a_i\}$ where $a_i = \text{Tr}(\alpha^i), i = 0, 1, \dots$ which belongs to the set $\{-1, 15, -17\}$, the Hadamard transform spectrum belonging to the set $\{0, \pm 16\}$ and linear span 35.

B. Boolean Function Aspects of the WG transformation:

Using the polynomial basis $(1, \alpha, \alpha^2, \alpha^3, \alpha^4, \alpha^5, \alpha^6)$, where $\alpha^7 + \alpha + 1 = 0$, the algebraic normal form of the Boolean function that corresponds to $f(x)$ is given by

$$\begin{aligned} & x_0 \oplus x_1 x_3 \oplus x_0 x_1 x_3 \oplus x_2 x_3 \oplus x_0 x_2 x_3 \oplus x_1 x_2 x_3 \oplus x_0 x_3 x_4 \oplus x_2 x_3 x_4 \oplus x_0 x_5 \oplus x_0 x_1 x_5 \oplus \\ & x_0 x_2 x_5 \oplus x_1 x_2 x_5 \oplus x_3 x_5 \oplus x_1 x_3 x_5 \oplus x_1 x_4 x_5 \oplus x_2 x_4 x_5 \oplus x_2 x_3 x_4 x_5 \oplus x_1 x_6 \oplus x_2 x_6 \oplus \\ & x_0 x_2 x_6 \oplus x_0 x_3 x_6 \oplus x_4 x_6 \oplus x_0 x_4 x_6 \oplus \\ & x_2 x_4 x_6 \oplus x_1 x_3 x_4 x_6 \oplus x_0 x_5 x_6 \oplus x_1 x_5 x_6 \oplus x_1 x_2 x_5 x_6 \oplus x_0 x_3 x_5 x_6. \end{aligned}$$

Changing the basis using the following basis conversion matrix

$$\begin{pmatrix} \beta_0 \\ \beta_1 \\ \beta_2 \\ \beta_3 \\ \beta_4 \\ \beta_5 \\ \beta_6 \end{pmatrix} = \begin{pmatrix} 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 \\ \alpha \\ \alpha^2 \\ \alpha^3 \\ \alpha^4 \\ \alpha^5 \\ \alpha^6 \end{pmatrix}$$

we obtain

$$x_0 \oplus x_1 \oplus x_2 \oplus x_3 \oplus x_0 x_3 \oplus x_0 x_1 x_3 \oplus x_2 x_3 \oplus x_0 x_2 x_3 \oplus x_1 x_2 x_3 \oplus x_3 x_4 \oplus x_1 x_3 x_4 \oplus$$

$$x_2x_3x_4 \oplus x_5 \oplus x_0x_5 \oplus x_1x_5 \oplus x_1x_3x_5 \oplus x_0x_4x_5 \oplus x_3x_4x_5 \oplus x_2x_3x_4x_5 \oplus x_6 \oplus x_0x_6 \oplus x_0x_2x_6 \oplus x_1x_2x_6 \oplus x_0x_3x_6 \oplus$$

$$x_1x_3x_6 \oplus x_2x_4x_6 \oplus x_1x_3x_4x_6 \oplus x_5x_6 \oplus x_0x_5x_6 \oplus x_1x_5x_6 \oplus x_2x_5x_6 \oplus x_1x_2x_5x_6 \oplus x_3x_5x_6 \oplus x_0x_3x_5x_6 \oplus x_2x_3x_5x_6 \oplus x_3x_4x_5x_6$$

which is 1-resilient function with nonlinearity 56, algebraic degree 4 and linear span 35.

7 Conclusion

We provide a table which contains the profiles that we obtained in previous sections as a conclusion of this paper.

Table 1. Profiles of WG transformations

| WG Sequences profile | WG Sequence | WG Trans. as Boolean Func. | WG Trans. Boolean Profile |
|---|------------------------------------|---------------------------------|---|
| $2^n - 1$ | Period | \leftrightarrow Boolean | n variables |
| Yes | Balance | \leftrightarrow Balance | Yes |
| Yes | 2-tuple distribution | NC | |
| 2-level | Auto correlation | NC | |
| $\{-1, -1 \pm 2^{\frac{n+1}{2}}\}$, n odd optimal w.r.t. the Welch bound, | cross correlation with m-sequences | \leftrightarrow Non-linearity | $2^{n-1} - 2^{\frac{n-1}{2}}$, n odd |
| $0, \pm 2^{(n+1)/2}$, n odd | Hadamard transform spectrum | | $0, \pm 2^{(n+1)/2}$, n odd |
| $n(2^{\lceil n/3 \rceil} - 3)$ increases exponentially in n | Linear span | \leftrightarrow Linear span | $n(2^{\lceil n/3 \rceil} - 3)$ increases exponentially in n |
| | NC | Degree | $\lceil n/3 \rceil + 1$ |
| | NC | r -resilient | $r : 1 \leq r \leq n - \lceil n/3 \rceil$ |
| easy* | Implementation | | easy |
| Ideal candidates for combining functions Pseudo-random sequence generators | Applications | | Ideal candidates for combining functions or filtering functions operating on a set of LFSRs |

Notations used in Table 1:

- NC means that there is no corresponding concept between them.
- *: There are two methods to implement WG sequences. One is to use 5 LFSRs together with a table look-up (Lemma 1 method). The other is to use a finite field configuration. The complexity of implementation of WG sequences by using the finite field configuration only depends on evaluation of four exponentiations listed in Section 2. Especially, it only depends on the evaluating the exponents q_3 and q_4 where each of them has $k-1$ consecutive 1's. We will discuss how to efficiently compute these two exponentiations at a separate paper. (*Note.* Implementation of the trace function has no cost.)

References

1. Xingong Chang, Zongduo Dai and Guang Gong, Some cryptographic properties of exponential functions, *Advances in Cryptology -AsiaCrypt'94*, Lecture Notes in Computer Science, No. 917, Springer-Verlag, 1994, pp. 415-418.

2. John Dillon, Multiplicative difference sets via additive characters, *Designs, Codes and Cryptography*. **17** (1999), no. 1-3, 225–235.
3. H. Dobbertin, Kasami power functions, permutation polynomials and cyclic difference sets, *Proceedings of the NATO-A.S.I. Workshop "Difference sets, sequences and their correlation properties"*, Bad Windsheim, August 3-14, 1998, Kluwer, Dordrecht, pp. 133-158, 1999.
4. J. Dillon and H. Dobbertin, New cyclic difference sets with Springer parameters, preprint, August 1999.
5. S.W. Golomb, *Shift Register Sequences*, Revised Edition, Aegean Park Press, 1982, pp. 39.
6. G. Gong, P. Gaal and S.W. Golomb, A suspected infinity class of cyclic Hadamard difference sets, *the Proceedings of 1997 IEEE Information Theory Workshop*, July 6-12, 1997, Longyearbyen, Svalbard, Norway.
7. G. Gong and A. M. Youssef, On Welch-Gong Transformation Sequence Generators, *Technical report, University of Waterloo, CORR 2000-30, May 2000*, <http://www.cacr.math.uwaterloo.ca/techreports/2000>.
8. R. Gold, Maximal recursive sequences with 3-valued recursive cross-correlation functions, *IEEE Trans. on Inform. Theory*, January 1968, pp. 154-156.
9. G. Gong, Lecture notes, <http://cacr.math.uwaterloo.ca/~ggong/CO739x/739xcover.html>
10. G. Gong and S.W. Golomb, Transform domain analysis of DES, *IEEE Trans. on Inform. Theory*, vol. 45, No.6, September 1999, pp. 2065-2073.
11. R. Lidl and H. Niederreiter, *Finite Fields*, Encyclopedia of Mathematics and its Applications, Volume 20, Addison-Wesley, 1983.
12. J.S. No, S.W. Golomb, G. Gong, H.K. Lee, and P. Gaal, New binary pseudo-random sequences of period $2^n - 1$ with ideal autocorrelation, *IEEE Trans. on Inform. Theory*, vol. 44, No. 2, March 1998, pp.814-817.
13. F.J. MacWilliams and N.J.A. Sloane, *The Theory of Error-Correcting Codes*, North-Holland Publishing Company, 1977.
14. A. Maschietti, Difference sets and hyperovals, *Designs, Codes and Cryptography*, **14**, pp. 89-98, 1998.
15. J. M. Massey, *Shift register synthesis and BCH decoding*, IEEE transactions on Information Theory. Vol. 15, no. 1, pp. 122-127, January, 1969.
16. Jong-Seon No, Hagong Chung and Min-Seon Yin, Binary pseudo-random sequences of period $2^m - 1$ with ideal autocorrelation generated by the polynomial $z^d + (z+1)^d$, *IEEE Trans. Inform. Theory*, vol. 44, no. 3, 1998, pp.1278-1282.
17. R.A. Rueppel, *Analysis and Design of Stream Ciphers*, Springer-Verlag, 1986, pp.53.
18. T. Siegenthaler, Correlation-immunity of nonlinear combining functions for cryptographic applications, *IEEE Trans. on Inform. Theory*, vol. IT-30, No. 5, September 1984, pp.776-780.
19. A.F. Webster and S.E. Tavares, On the design of S-boxes, *Advances in Cryptology-Crypto'85*, Lecture Notes in Computer Science, No. 218, Springer-Verlag, 1985, pp. 523-534.
20. Guozheng Xiao and James Massey, A spectral characterization of correlation immune combining functions, *IEEE Trans. on Inform. Theory*, vol. 34, No. 3, May 1988, pp.569-571.
21. A. M. Youssef and G. Gong, On the interpolation attacks on block ciphers, *Proceedings of Fast Software Encryption 2000*, April 13-14, 2000, New York, Lecture Notes in computer science, to appear.

Modes of Operation of Stream Ciphers

Jovan Dj. Golić*

School of Electrical Engineering, University of Belgrade,
Bulevar Revolucije 73, 11001 Beograd, Yugoslavia
`golic@galeb.etf.bg.ac.yu`

Abstract. A general stream cipher with memory in which each ciphertext symbol depends on both the current and previous plaintext symbols, as well as each plaintext symbol depends on both the current and previous ciphertext symbols, is pointed out. It is shown how to convert any keystream generator into a stream cipher with memory and their security is discussed. It is proposed how to construct secure self-synchronizing stream ciphers, keyed hash functions, hash functions, and block ciphers from any secure stream cipher with memory. Rather new and unusual designs can thus be obtained, such as the designs of block ciphers and (keyed) hash functions based on clock-controlled shift registers only.

Key words: Stream ciphers, block ciphers, keyed hash functions, hash functions, conversions, security.

1 Introduction

The electronic codebook (ECB) mode of block ciphers is mostly confined to encryption of relatively short messages in cryptographic protocols for confidentiality and/or authentication purposes. To increase the resistance to cryptanalysis, various modes with memory have been suggested, such as the output feedback (OFB) mode, the cipher block chaining (CBC) mode, the cipher feedback (CFB) mode, and the counter mode (e.g., see [22]). If a block cipher is used for encryption in one of these modes with memory, it then essentially becomes a stream cipher whose next-state and/or output functions are determined by the secret-key-dependent encryption and/or decryption functions of the block cipher.

The CBC and CFB as well as some other modes with memory [20] can be used to produce a message authentication code (MAC), usually also called a keyed hash function (KHF), which can be combined with encryption as well. It is proved in [2] that the CBC-MAC (used without CBC encryption) is at least as secure as the underlying block cipher with respect to the well-known pseudorandom function probabilistic model. A more general theoretical framework for the relative security analysis of symmetric encryption modes is proposed in [3]. Block ciphers can also be used in a number of various modes with memory to build hash functions (HF's) (e.g., see [20]).

* Part of this work was done while the author was with the Information Security Research Centre, Queensland University of Technology, Brisbane, Australia

However, stream ciphers need not be based on block ciphers, that is, on one-to-one functions that are difficult to compute in both directions without knowledge of the secret key. Instead, there exist many proposals of practically secure stream ciphers (keystream generators) whose next-state and/or output functions are very simple and whose initial state is controlled by the secret key (e.g., see [21], [22], and [19]).

A general way to produce a KHF from a keystream generator and an unconditionally secure MAC (authentication code, which is typically based on some linear operations modulo an integer or modulo a polynomial) is to use the keystream sequence to define the time-variant MAC secret key (see [23], [12], and [14]). A different construction directly based on a keystream generator is given in [15], but is shown to be insecure in [23]. The KHF can be combined with the keystream generator encryption, but then an additional portion of the keystream sequence is required.

In [1] it is suggested how to build block ciphers of unbalanced Luby-Rackoff type, with just a few rounds, from a keystream generator and a KHF or a HF. Note that a KHF itself can be produced from a HF by incorporating the secret key in the message. Similar constructions are also proposed in [16], along with a more elaborate security analysis.

The main objective of this paper is to show how to construct secure self-synchronizing stream ciphers, keyed hash functions, hash functions, and block ciphers from secure stream ciphers in a general, simple, and direct way. The crucial point is that instead of the keystream generator mode, which is almost exclusively treated in the open literature, we make use of a more general stream cipher mode in which each ciphertext symbol depends not only on the current plaintext symbol, but also on all the previous plaintext symbols. We discuss the security of all the modes proposed.

A formal proof that a derived mode of operation is at least as secure as the original mode with respect to certain attacks means that any efficient attack on the derived mode gives rise to an efficient attack on the original mode. However, if the original mode is not proved to be secure itself, as is the case with all known practical stream and block ciphers and (keyed) hash functions in the open literature, then formal security reduction proofs only show alternative ways of developing efficient attacks on the original mode. If the original mode is considered heuristically secure with respect to known attacks, then formal security reduction proofs do not imply that the derived modes are such, because they may be vulnerable to previously unknown, specially adapted attacks. Moreover, in our case, the security of the underlying stream cipher mode with plaintext memory has not been considered in the open literature. Accordingly, apart from the formal security analysis, we also deal with the practical (heuristic) security analysis of the modes proposed.

As usual (e.g., see [18] and [1]), the formal security statements and proofs are presented in a general language, which can be made mathematically precise by assuming various mathematical models for cryptanalytic attacks, such as the polynomial-complexity algorithms.

The stream cipher with memory (SCM) and other stream cipher modes are briefly described in Section 2. A general way of converting a stream cipher from the keystream generator mode into the SCM mode is proposed in Section 3 and practical security analysis of both the modes is addressed in Section 4. Conversions of a stream cipher with memory into a self-synchronizing stream cipher, a keyed hash function, a block cipher, and a hash function are presented in Sections 5, 6, 7, and 8, respectively, together with their security analyses. A proposal of a simple class of stream ciphers with memory to be used is described in Section 9. Conclusions are given in Section 10.

2 Stream Cipher with Memory (SCM) Mode

Let $x = (x_t)_{t=0}^{\infty}$ and $y = (y_t)_{t=0}^{\infty}$ denote the plaintext and ciphertext binary sequences, respectively. Let the binary strings k and r stand for the secret and randomizing keys, respectively, and let $s = (s_t)_{t=0}^{\infty}$ denote the internal state sequence where s_t is a binary string of length M and $s_0(k, r)$ is the initial internal state determined by k and r . Then, a general binary stream cipher decipherable without delay is an invertible nonautonomous finite-state machine with one input and one output that maps an input sequence x into the output sequence y by the encryption (sequential) transform recursively defined by

$$s_{t+1} = F_k(s_t, x_t), \quad y_t = x_t + f_k(s_t), \quad t \geq 0 \quad (1)$$

where the addition is binary and $F_k : \{0, 1\}^{M+1} \rightarrow \{0, 1\}^M$ and $f_k : \{0, 1\}^M \rightarrow \{0, 1\}$ are the (secret-key-dependent) next-state and output functions, respectively. The inverse, decryption transform is recursively defined by

$$s_{t+1} = F_k(s_t, y_t + f_k(s_t)), \quad x_t = y_t + f_k(s_t), \quad t \geq 0. \quad (2)$$

The encryption and decryption transforms are thus defined by the so-called keystream sequence $z = (z_t = f_k(s_t))_{t=0}^{\infty}$.

Let the input memories of the encryption and decryption transforms be referred to as the input and output memories of a stream cipher, respectively. All stream ciphers can be classified into the following three types with respect to the input and output memories: memoryless, with finite input or output memory, and with infinite input and output memory.

In the memoryless type, known as the keystream generator (KG) or the pseudorandom sequence generator type, the next-state function does not depend on the plaintext symbol, that is, $s_{t+1} = F_k(s_t)$, so that the keystream sequence z is plaintext independent. The KG type is sensitive to synchronization errors but has no substitution error propagation. To deal with a possible loss of synchronization when encrypting longer messages with the same secret key k , a long message is divided in shorter ones and a randomizing key r is used to reinitialize the keystream generator for every new message to be encrypted. It is typically sent in the clear and as such is public rather than secret. The randomizing key is generated in a random or deterministic way with the property of being with

high probability different for every new message to be encrypted with the same secret key. This is in order to satisfy the so-called *one-time-pad assumption* that repetitions of long segments of keystream are highly unlikely. Namely, to this end, every new plaintext sequence should with high probability be encrypted by using a different initial state. More generally, repetitions of internal states should be highly unlikely.

In the finite input or output memory type, the encryption or decryption transform has finite input memory. In particular, in the self-synchronizing stream cipher (S²SC) type (i.e., the cipher feedback type), the decryption transform has finite input memory, that is, $s_{t+1} = (y_{t-i})_{i=0}^{M-1}$, so that the keystream sequence depends on ciphertext only. As a consequence, the propagation of both synchronization and substitution errors in decryption is limited. Its security entirely depends on the output (feedback) function f_k which must be secret-key-dependent.

In the infinite input and output memory type, the next-state function effectively depends on the current plaintext symbol in such way that both the encryption and decryption transforms have infinite input memory. This type is typically either not mentioned or just overlooked as a practical possibility in the open literature on stream ciphers (e.g., see [21], [22], and [19]). Such a type is called here the stream cipher with memory (SCM) type, to emphasize the fact that in encryption each ciphertext bit does not depend on the current plaintext bit only, but also on the previous plaintext bits as well as that in decryption each plaintext bit depends on the current and previous ciphertext bits. Note that the PKZIP stream cipher is of this type (see [5]).

The SCM type is therefore sensitive to both synchronization and substitution errors, but, due to infinite input memory, has an inherent potential that can be used for message integrity purposes. The errors on real channels should be dealt with by separate error-correction and/or detection codes and, in addition, by the resynchronization method. However, instead of using and transmitting the randomizing key as described above, one may just prepend the randomizing key to each new message and keep the initial state secret-key-dependent only, as is done in the PKZIP stream cipher. In this case, the randomizing key is encrypted rather than transmitted in the clear and hence need not be public.

The basic types of stream ciphers will also be referred to as the modes of operation of stream ciphers, because it will be shown that they can be converted into each other by simple and general constructions. Similarly, other cryptographic primitives constructed from stream ciphers will also be referred to as the modes of operation of stream ciphers.

3 Conversion of Keystream Generator (KG) Mode into SCM Mode

Any stream cipher in the KG mode can be converted into the SCM mode by letting the next-state function depend upon the current plaintext bit too. The main practical criterion to be respected in this regard is that a change of a single plaintext bit should give rise to a random looking change in the keystream

(ciphertext) sequence to follow (forward propagation effect). Note that the KG mode should satisfy the same property, but with respect to changes of the initial state bits. So, the adaptation is easily achieved by adding the plaintext bit to one or more of the internal state bits, especially those with the significant forward propagation effect. For a nonbinary plaintext alphabet, the plaintext symbol should be incorporated by a function with the property that any change of this symbol necessarily gives rise to a change of one or more of the internal state variables. This can generally be achieved by using a quasigroup operation.

In shift-register-based keystream generators, the conversion is easily done by making the shift registers nonautonomous, that is, by adding the plaintext bit to the feedback bit for each of the shift registers, and especially those that are used for clock control. The output of such shift registers should then necessarily involve the first, input stage. In KGs based on the table-shuffling principle like RC4 (see [22]), the plaintext symbol should affect the internal state variables defining the table positions where the changes should be made.

4 Security Analysis of SCM and KG Modes

In view of (1), the change of one plaintext bit necessarily causes the change of the corresponding ciphertext bit in the KG and SCM modes as well as a pseudorandom change of only the subsequent ciphertext bits in the SCM mode. Therefore, without the one-time-pad assumption, cryptanalytic attacks generating new plaintext/ciphertext pairs by modifying some bits in the known pairs are feasible. This is relevant for achieving information authenticity. The one-time-pad assumption can be achieved either without using resynchronization or by using resynchronization by (with high probability) different randomizing keys.

We will discuss cryptanalytic attacks in a general, adaptive combined chosen plaintext and ciphertext scenario, possibly key related as well (e.g., see [18]). The situation is conceptually similar to one with block ciphers in the ECB mode, with a difference that we now deal with the plaintext/ciphertext strings rather than blocks and that each plaintext sequence effectively includes the randomizing key used. The attacks use a training set consisting of a number of known, arbitrarily, possibly adaptively, chosen plaintext/ciphertext string pairs obtained from the same secret key (and the same or different known randomizing keys), and possibly from a set of related keys as well. An attack is an algorithm that produces one or more new plaintext/ciphertext string pairs, not included in the training set, where either the plaintext or ciphertext string in each of these pairs is assumed to be given. As knowing a plaintext/ciphertext string pair in the KG mode is equivalent to knowing a keystream/ciphertext string pair, an attack on the KG mode is in fact an algorithm that reconstructs unknown portions of a keystream sequence from its known portions or, more generally, from known portions of a set of keystream sequences obtained from different randomizing keys and the same secret key.

Typically, it is assumed that the attacks work for any secret key. Ideally, the exhaustive search over the secret keys or over unknown plaintext or ciphertext

strings should be the most efficient way to perform an attack. If an attack can recover only particular plaintexts from given ciphertexts, it is then said to be successful for such plaintexts only, e.g., corresponding to certain plaintext statistics. On the other hand, secret key reconstruction attacks are the best known examples of the attacks successful for all the plaintexts.

An attack on the SCM mode that produces new plaintext/ciphertext string pairs by modifying a number of end bits in a known plaintext/ciphertext string pair is said to be *trivial*. A trivial attack on the KG mode is defined similarly. As argued above, trivial attacks are always computationally feasible for any stream cipher in the SCM or KG mode if the one-time-pad assumption is not satisfied.

A stream cipher in the SCM or KG mode is said to be (practically) secure with respect to nontrivial cryptanalytic attacks if no such attack is computationally feasible, relative to available computing power. This security essentially means that both the encryption and decryption transforms are infeasible to compute without knowing the secret key, under the one-time-pad assumption.

Since it does not appear possible to formally relate the security of the KG and SCM modes with respect to general attacks, we will concentrate on their practical security. It turns out that the security of the SCM mode is more related to the security of the KG mode with than without resynchronization, which, except for [6], is hardly analyzed in the open literature.

The fact that in the SCM mode the plaintext statistics is disguised by the plaintext dependent keystream sequence makes the cryptanalytic attacks successful for particular plaintexts only and the ciphertext-only cryptanalytic attacks both very unlikely. For the same reason, the attacks consisting in predicting the keystream without reconstructing the initial state (e.g., based on low linear [21] or 2-adic [13] complexities) and the attacks finding the statistical weaknesses of the keystream (e.g., [9]) are also unlikely to succeed. In addition, unlike the KG mode, assuming that the (prepended) randomizing key is known is not very realistic in the SCM mode. Also, the known plaintext/ciphertext scenario is generally less useful for the SCM than for the KG mode, as missing portions of the plaintext/ciphertext sequences present a difficulty which is harder to overcome in the SCM than in the KG mode.

On the other hand, the plaintext dependent keystream sequence may open new possibilities for secret key reconstruction attacks especially if resynchronization is used and if prepended resynchronization key is assumed to be known, although the cryptanalysis is more complicated. For a survey of various (initial state) secret key reconstruction attacks on the KG mode without resynchronization, see [21], [7], [10], and [19].

Since the randomizing key plays the role of known plaintext, the cryptanalytic methods for block ciphers in the ECB mode, such as the differential [4] and linear [17] cryptanalysis, in principle extend to the KG and SCM modes too, especially if the secret and randomizing keys are linearly combined together. They are less likely to succeed here because of the underlying iterative structure. On the other hand, if long all zero plaintext sequences are allowed to be encrypted, then, formally, any secret key reconstruction attack on the KG mode directly

extends to the SCM mode. So, the basic criterion for the SCM mode is that the underlying KG mode is secure (e.g., this is not true for the PKZIP stream cipher cryptanalyzed in [5]).

If the prepended randomizing key is assumed to be known, then the SCM mode is required to be secure with respect to nontrivial cryptanalytic attacks (in particular, secret key reconstruction attacks), for any training set of known plaintext/ciphertext sequence pairs produced from the same initial state. To this end, when the plaintext symbol is introduced into a part of the next-state function of the underlying KG mode by a linear function, it appears reasonable to recommend that this part of the next-state function should not be linear, as a whole, with respect to the same type of linearity. Accordingly, in binary shift-register-based keystream generators at least one of the shift registers affected by the plaintext bit should be clock controlled or have nonlinear feedback.

5 Conversion of SCM Mode into Self-Synchronizing Stream Cipher (S²SC) Mode

We should define the secret-key-controlled feedback function f_k of m binary variables, where m is the output memory size, on the basis of the SCM mode of a given stream cipher so as to prevent trivial attacks on the SCM mode. A general and simple way to achieve this is to use an SCM mode with the secret key k , with the initial state determined only by k , and with the plaintext sequence whose first m plaintext bits are defined by a given m -bit input to f_k and the remaining plaintext bits are fixed, possibly to zero. The output bit of f_k is then defined as the last ciphertext bit obtained after clocking the SCM mode m times and a specified additional number of times, typically, on the order of several (e.g., three) internal memory sizes M of the SCM mode. Similarly, a faster conversion is obtained by producing a block of n , $n \leq M$, ciphertext bits at a time, thus constructing an n -bit output feedback function f_k . In the S²SC mode, its output is then combined with an n -bit block of plaintext at a time, e.g., by using the bitwise binary addition.

Any attack on the S²SC mode is essentially an algorithm for producing the unknown outputs of the feedback function for one or more given inputs, where a training set of a number of input/output pairs of the feedback function is assumed to be known. The S²SC mode of a stream cipher is said to be secure if no such attack is computationally feasible. In particular, the security can be defined with respect to secret key reconstruction attacks only.

Proposition 1. *If the underlying SCM mode is secure with respect to nontrivial cryptanalytic attacks, then the derived S²SC mode is secure. If the underlying SCM mode is secure with respect to secret key reconstruction cryptanalytic attacks, so is the derived S²SC mode.*

Proof. It should be proved that any computationally feasible attack on the S²SC mode can be converted into a nontrivial attack on the SCM mode at an additional computational cost that is not comparatively significant. This is a direct

consequence of the proposed construction, as any input/output pair for the feedback function can be extended into a plaintext/ciphertext string pair for the SCM mode by appending a required number of fixed plaintext bits to the input bits, where only the last bit (or the last n bits) of ciphertext is known. Accordingly, one can modify the training set for any attack on the S²SC mode into the corresponding training set for an attack on the SCM mode and vice versa. As well, producing an unknown input/output pair of the feedback function directly extends into producing an unknown bit (or n bits) of ciphertext for a known plaintext string of the SCM mode. This is by definition a nontrivial attack on the SCM mode, since sufficiently many fixed plaintext bits are appended.

A similar proof holds for secret key reconstruction attacks. \square

In other words, the derived S²SC mode is at least as secure as the underlying SCM mode with respect to nontrivial attacks, as trivial attacks on the SCM mode are prevented by additional clocking.

6 Conversion of SCM Mode into Keyed Hash Function (KHF) Mode

A binary keyed hash function (KHF) or a message authentication code (MAC) is a secret-key-dependent function $\{0, 1\}^l \rightarrow \{0, 1\}^n$ that maps binary strings of variable length l (messages) into binary strings of a fixed length n , where $l \geq n$. This function should be easy to compute when the secret key is known. The main computational security property required from a KHF is that it should be computationally infeasible, without knowledge of k , to perform existential forgery in the (adaptive) chosen message scenario, that is, to find another, distinct message and its hash value under k , provided a set of (adaptively) chosen messages and their hash values under k is given. In particular, a KHF should be secure against any secret key reconstruction attack. Ideally, the exhaustive search over k should be the most efficient way to find k .

Our objective now is to show how to construct a KHF from a stream cipher in the SCM mode. The designs proposed in the literature are either dedicated or are based on block ciphers, hash functions, or unconditionally secure MACs combined with KGs. They typically require that the message length be a multiple of a given positive integer which is achieved by padding. Our construction allows an arbitrary message length and is solely based on a stream cipher in the SCM mode, which can be easily obtained from any KG mode (as shown in Section 3).

The construction is similar to the one proposed for the S²SC mode. Let k be the secret key for the SCM mode of a given stream cipher with the initial state determined only by k and with the plaintext sequence whose first l plaintext bits are defined by the given message and the remaining bits are fixed, possibly to zero. Let $M \geq n$, where M is the internal memory size of the SCM mode. The hash value is then defined as the last n successive ciphertext bits obtained after clocking the SCM mode l times and a specified additional number of times, several (e.g., three) times bigger than M .

In a similar way as for the S²SC mode, producing a message and the corresponding hash value under k for the defined KHF mode directly extends into producing n unknown ciphertext bits for a known plaintext string of the underlying SCM mode. Consequently, the following proposition is proved in essentially the same way as Proposition 1.

Proposition 2. *If the underlying SCM mode is secure with respect to nontrivial cryptanalytic attacks, then the derived KHF mode is secure. If the underlying SCM mode is secure with respect to secret key reconstruction cryptanalytic attacks, so is the derived KHF mode.*

In other words, the derived KHF mode is at least as secure as the underlying SCM mode with respect to nontrivial attacks, as trivial attacks on the SCM mode are prevented by additional clocking. The same proposition holds even if the hash value of a given message is used together with the corresponding ciphertext obtained by the SCM with the same k . Accordingly, the KHF mode can be combined with the SCM encryption with the same secret key. Additional protection measures or constraints typically required for the designs proposed in the literature (e.g., due to the finite input memory of the CBC or CFB decryption) are not here needed.

7 Conversion of SCM Mode into Block Cipher (BC) Mode

A binary block cipher is a secret-key-dependent one-to-one function $\{0, 1\}^n \rightarrow \{0, 1\}^n$ that maps binary strings of length n (plaintext blocks) into binary strings of the same length (ciphertext blocks), where the block length n is usually fixed and relatively short. The function is called the encryption function, and its inverse is called the decryption function. Both the functions should be easy to compute when the secret key k is known and infeasible to compute when k is not known, in the (adaptive) chosen plaintext/ciphertext scenario, possibly key related as well. More precisely, an attack on a block cipher is an algorithm that, on the basis of a training set consisting of a number of known, arbitrarily chosen plaintext/ciphertext block pairs, produces one or more new plaintext/ciphertext block pairs, where either plaintext or ciphertext blocks are assumed to be given. A block cipher is said to be secure with respect to cryptanalytic attacks if no such attack is computationally feasible.

Typically, the objective of cryptanalytic attacks on block ciphers is to reconstruct the secret key, in which case the attacks are successful for all the plaintexts. The differential cryptanalysis [4] in the chosen plaintext scenario and the linear cryptanalysis [17] in the known plaintext scenario are the well-known examples. The vast majority of existing proposals for block ciphers use the product structure composed of a number of rounds each involving a relatively simple one-round function, such as the Feistel type ciphers like DES.

We will now describe a simple and general way to construct a secure block cipher (BC) mode starting from any secure stream cipher in the SCM mode.

The construction essentially requires only three rounds and works for variable block sizes that are practically limited only by the memory space available. We are interested in the SCM mode whose initial state depends on the secret key only, without using any randomizing key to satisfy the one-time-pad assumption. Such a mode is not secure with respect to the so-called trivial attacks, as it is possible to generate, with high probability of success, new plaintext/ciphertext string pairs by modifying a number of end bits in the plaintext/ciphertext string pairs already known. So, what is essentially needed is a construction that would prevent such attacks.

We propose a product connection of three stream ciphers in the same SCM mode whose secret keys are the same, independent, or different (but related). Product ciphers with independent keys are usually called cascade ciphers (see [18]). In fact, we suggest different keys related in a very simple way, which removes the need for a special key schedule. For example, the keys for the second and the third cipher can be obtained as cyclic shifts of the key for the first stream cipher (represented as a binary string). The main point is to use the output bits from the first/second stream cipher in the reverse order to define the input bits for the second/third stream cipher, respectively. This creates the required forward propagation effect for the second half of the plaintext bits, as noted in [11] in a different context of block ciphers based on specific finite automata. It is required to memorize the outputs of the first and the second stream cipher. One can also use a product connection of only two stream ciphers, but in this case the change of the last plaintext bit necessarily gives rise to the change of the first ciphertext bit, which may be considered as a weakness for some applications.

For the encryption of long messages, the proposed BC mode can be used in its basic, ECB form with a large block size. Also, it can be used in the usual CBC and CFB modes, with a finite input memory of the decryption transform. This may be suitable for some applications, e.g., for the encryption of random access files. In addition, one may also use the existing internal memory of the underlying stream cipher. For example, one can use the last generated internal state for the current plaintext block as the initial state for the next one. The obtained cipher can then be considered as an enhanced version of the original stream cipher in the SCM mode, because of the triple encryption.

By using the standard argument (e.g., see [18]), we obtain the following two propositions, which essentially show that the BC mode is at least as secure as the underlying SCM mode with respect to nontrivial attacks, as trivial attacks on the SCM mode are prevented by reversing the intermediate ciphertexts. It is assumed that the attacks work for any secret key.

Proposition 3. *If the underlying SCM mode is secure with respect to nontrivial cryptanalytic attacks, then the derived BC mode with the cascade connection is secure. If the underlying SCM mode is secure with respect to secret key reconstruction cryptanalytic attacks, so is the derived BC mode with the cascade connection.*

Proof. It should be proved that any computationally feasible attack on the BC mode with the cascade connection can be converted into a nontrivial attack on

the SCM mode at an additional computational cost that is not comparatively significant. Consider the BC mode where the secret key for the first stream cipher is unknown and arbitrary and where the keys for the second and the third stream cipher are known and fixed. Then any plaintext/ciphertext block pair for the BC mode can be converted into the corresponding plaintext/ciphertext string pair for the SCM mode of the first stream cipher and vice versa, at a computational cost of two SCM encryptions/decryptions (with known keys) per pair.

Accordingly, one can modify the training set for any attack on the BC mode into the corresponding training set for an attack on the SCM mode and vice versa. Also, producing an unknown plaintext/ciphertext block pair for the BC mode directly extends into producing an unknown ciphertext string for a known plaintext string for the SCM mode. This is a nontrivial attack on the SCM mode, since the reversion of intermediate ciphertexts renders the produced plaintext/ciphertext string pair(s) different from those obtained by trivial attacks (i.e., by modifying pairs from the training set).

A similar proof holds for secret key reconstruction attacks. \square

Proposition 4. *If the underlying SCM mode is secure with respect to secret key reconstruction cryptanalytic attacks in the related key scenario, then the derived BC mode with the product connection is secure with respect to secret key reconstruction cryptanalytic attacks.*

Proof. It should be proved that any computationally feasible secret key reconstruction attack on the BC mode with the product connection can be converted into a secret key reconstruction attack on the underlying SCM mode at an additional computational cost that is not comparatively significant. As the keys of the second and the third SCM mode are derived from the secret key of the first SCM mode, any given plaintext/ciphertext block pair for the BC mode can be converted into the corresponding plaintext/ciphertext string pair for the first SCM mode and vice versa if two more plaintext/ciphertext string pairs are known: one for the second SCM mode and one for the third SCM mode in the product connection, both obtained from the (related) keys derived from the secret key of the first SCM mode.

Accordingly, the training set for any secret key reconstruction attack on the BC mode can be obtained from the corresponding training set for the SCM mode with the same secret key and from two additional training sets for the SCM modes with related keys. This is achieved by combining the plaintext/ciphertext string pairs from the three training sets into the corresponding plaintext/ciphertext block pairs for the BC mode, respectively. The secret key for the first SCM mode can then be produced by the secret key reconstruction attack on the corresponding BC mode. \square

Note that in a special case, Proposition 3 is true for cryptanalytic attacks successful for particular plaintexts only, as the plaintexts for the first SCM mode in the cascade and for the whole cascade are the same (see [18]). If very simple stream ciphers are used so that the security of the SCM mode may be questionable, then the number of rounds can be increased.

8 Conversion of SCM Mode into Hash Function (HF) Mode

A binary hash function (HF) is defined in the same way as a KHF except that the secret key parameter is not used. One-wayness and collision-resistance (or collision-freedom) are the two main computational security properties required from a HF. A HF is called one-way if it is computationally infeasible to find any input that hashes to a given output, for almost all outputs. Ideally, the random guessing, with the computational complexity $O(2^n)$, should be the most efficient way of inverting a HF. A HF is called collision-resistant if it is computationally infeasible to find any two distinct inputs that hash to the same output. Ideally, the birthday attack, with the computational complexity $O(2^{n/2})$, should be the most efficient way of producing collisions. Note that the existing proposals for HF's are either dedicated or are based on block ciphers like DES, and are typically defined in terms of a so-called compression function which is applied iteratively (e.g., see [20] and [1]).

By fixing the value of the secret key, any KHF obtained from the SCM mode of a given stream cipher becomes a candidate HF. However, its security is no longer guaranteed by the security of the SCM mode. Namely, the one-wayness and collision-resistance properties impose stronger requirements for the SCM mode which do not involve the output function of the SCM mode at all and are hence more difficult to satisfy. For example, collision-resistance implies that it should be computationally infeasible to find any two different plaintext sequences that will, starting from the same initial state, produce the same internal state at a given time in future.

We now define a more complicated, but still simple construction of a HF from the SCM mode whose security relies on the output function as well. The basic construction consists of two stages. The first stage is similar to one for a KHF, except that the plaintext sequence for the SCM mode consists of the l message bits only and that the (secret) key is fixed and known. The SCM is clocked l times and the corresponding l bits of the ciphertext are memorized. In the second stage, the l ciphertext bits in the reverse order are used as the plaintext sequence for the same SCM mode, but now starting from the last internal state produced in the first stage. The SCM is clocked l times and an additional number of times several times bigger than M (as before), and the last n successive ciphertext bits produced are the hash value.

As in the BC mode, the ciphertext bits are used in the reverse order to increase the forward propagation effect for the second half of the message bits. Since finding the collisions necessarily involves the output function of the SCM mode, the constructed HF seems to be, at least heuristically, at least as secure as the underlying SCM mode with respect to nontrivial cryptanalytic attacks. Clearly, the number of stages can be made bigger than two by proceeding in a similar way. This would increase the security.

If the underlying SCM mode is secure, then the resulting KHF and HF modes also satisfy a stronger security property that any change of the message bits gives rise to a random looking change of the corresponding hash value.

As the basic construction described above requires memorizing intermediate ciphertext(s) of the same length as the message itself, a derived construction with lesser memory requirements would be to use the basic construction to define the compression function which is applied iteratively in the usual way. Namely, let the message be divided into blocks of a given, relatively large length l with the last block of variable length as required (without any padding). Let, for simplicity, the memory size of the SCM mode utilized be equal to the hash value length n . Then each round of the iterative construction is the basic construction applied to a new message block and with the hash value from the previous iteration as the initial internal state.

9 Proposal

The underlying keystream generator to be used in the proposed constructions can be as simple as a self-clock-controlled nonlinear filter generator, where irregular clocking is needed to ensure that the next-state function is nonlinear. The nonlinear filter generator, when regularly clocked, should be designed so as to resist known initial state reconstruction attacks including the fast correlation attack, the conditional correlation attack, and the inversion attack as well as to achieve (with a high probability) a long period, a high linear complexity, and good statistical properties of the output sequence (see [8]). We propose that the LFSR length be at least 256 and that the LFSR initial state be defined by the secret key at least 128 bits long. As the next-state function is not one-to-one due to irregular clocking, one may expect a reduced period of the keystream sequence, but not less than about 2^{128} , which is long enough even for stream cipher applications.

The binary clock-control output is produced by an additional boolean function with a few inputs (e.g., three) taken from the LFSR taps chosen according to a full positive difference set. The difference sets used for clock control and for the filter function should be disjoint, as in a nonlinear filter generator with two binary outputs (see [8]). According to the binary clock-control output, the LFSR is clocked once or twice per each output bit. The SCM mode is formed by adding the plaintext bit to the feedback bit at each time, with the plaintext bit repeated if the LFSR is clocked twice.

10 Conclusions

A general stream cipher with memory (SCM) mode, which is typically overlooked in the open literature, is pointed out. Its main characteristic is that each ciphertext symbol depends on both the current and previous plaintext symbols. Similarly, in decryption, each plaintext symbol depends on the current and previous ciphertext symbols. It is shown how to convert any keystream generator (KG) mode into the SCM mode and their practical security is discussed. Investigating the practical security of the SCM mode of stream ciphers is a new interesting research area in public cryptography. Developing attacks on the SCM

mode would reveal weaknesses of the underlying KG mode, especially when used with resynchronization.

It is proposed how to obtain a secure self-synchronizing stream cipher from any secure stream cipher in the SCM mode. It is then proposed how to construct secure keyed hash functions, block ciphers, and hash functions from any secure stream cipher in the SCM mode. In all the modes, the message length can be made large and variable in a simple and natural way.

The resulting designs are rather new and unusual and are based on the iterative structure of stream ciphers which is symbol rather than block based. The way the secret key is incorporated is new too. For example, there is no need for specially designed S-boxes or a special key schedule algorithm. In particular, the underlying stream cipher can be as simple as a single self-clock-controlled nonlinear filter generator, with the secret key controlling its initial state only.

All the constructions directly extend from the binary to an arbitrary plaintext/ciphertext alphabet, e.g., to stream ciphers based on multiple rather than individual shift registers which are suitable for software realizations.

References

1. R. J. Anderson and E. Biham, "Two practical and provably secure block ciphers: BEAR and LION," Fast Software Encryption – Cambridge '96, *Lecture Notes in Computer Science*, vol. 1039, D. Gollmann ed., Springer-Verlag, pp. 113-120, 1996.
2. M. Bellare, J. Kilian, and P. Rogaway, "The security of cipher block chaining," Advances in Cryptology – CRYPTO '94, *Lecture Notes in Computer Science*, vol. 839, Y. G. Desmedt ed., Springer-Verlag, pp. 341-358, 1994.
3. M. Bellare, A. Desai, E. Jøkipii, and P. Rogaway, "A concrete security treatment of symmetric encryption: analysis of the DES modes of operation," *Proc. of the 38. Annual Symposium on the Foundations of Computer Science – FOCS '97*, IEEE Press, 1997.
4. E. Biham and A. Shamir, "Differential cryptanalysis of DES-like cryptosystems," *Journal of Cryptology*, vol. 4(1), pp. 3-72, 1991.
5. E. Biham and P. C. Kocher, "A known plaintext attack on the PKZIP stream cipher," Fast Software Encryption – Leuven '94, *Lecture Notes in Computer Science*, vol. 1008, B. Preneel ed., Springer-Verlag, pp. 144-153, 1995.
6. J. Daemen, R. Govaerts, and J. Vandewalle, "Resynchronization weakness in synchronous stream ciphers," Advances in Cryptology – EUROCRYPT '93, *Lecture Notes in Computer Science*, vol. 765, T. Hellesest ed., Springer-Verlag, pp. 159-167, 1994.
7. J. Dj. Golić, "On the security of shift register based keystream generators," Fast Software Encryption – Cambridge '93, *Lecture Notes of Computer Science*, vol. 809, R. J. Anderson ed., Springer-Verlag, pp. 90-100, 1994.
8. J. Dj. Golić, "On the security of nonlinear filter generators," Fast Software Encryption – Cambridge '96, *Lecture Notes in Computer Science*, vol. 1039, D. Gollmann ed., Springer-Verlag, pp. 173-188, 1996.
9. J. Dj. Golić, "Linear models for keystream generators," *IEEE Trans. Computers*, vol. C-45, pp. 41-49, Jan. 1996.
10. J. Dj. Golić, "Recent advances in stream cipher cryptanalysis," *Publications de l'Institut Mathématique*, vol. 64/78, pp. 183-204, 1998.

11. M. Gysin, "A one-key cryptosystem based on a finite nonlinear automaton," *Cryptography: Policy and Algorithms – Brisbane '95, Lecture Notes in Computer Science*, vol. 1029, E. Dawson and J. Golić eds., Springer-Verlag, pp. 165-173, 1996.
12. T. Johansson, "A shift register construction of unconditionally secure authentication codes," *Designs, Codes and Cryptography*, vol. 4, pp. 69-81, 1994.
13. A. Klapper and M. Goresky, "Cryptanalysis based on 2-adic rational approximation," *Advances in Cryptology – CRYPTO '95, Lecture Notes in Computer Science*, vol. 963, D. Coppersmith ed., Springer-Verlag, pp. 262-273, 1995.
14. H. Krawczyk, "LFSR-based hashing and authentication," *Advances in Cryptology – CRYPTO '94, Lecture Notes in Computer Science*, vol. 839, Y. G. Desmedt ed., Springer-Verlag, pp. 129-139, 1994.
15. X. Lai and R. A. Rueppel, "A fast cryptographic checksum algorithm based on stream ciphers," *Advances in Cryptology – AUSCRYPT '92, Lecture Notes in Computer Science*, vol. 718, J. Seberry and Y. Zheng eds., Springer-Verlag, pp. 339-348, 1993.
16. S. Lucks, "Faster Luby-Rackoff ciphers," *Fast Software Encryption – Cambridge '96, Lecture Notes in Computer Science*, vol. 1039, D. Gollmann ed., Springer-Verlag, pp. 189-203, 1996.
17. M. Matsui, "Linear cryptanalysis method for DES cipher," *Advances in Cryptology – EUROCRYPT '93, Lecture Notes in Computer Science*, vol. 765, T. Helleseth ed., Springer-Verlag, pp. 386-397, 1994.
18. U. M. Maurer and J. L. Massey, "Cascade ciphers: the importance of being first," *Journal of Cryptology*, vol. 6(1), pp. 55-61, 1993.
19. A. Menezes, P. van Oorschot, and S. Vanstone, *Handbook of Applied Cryptography*. Boca Raton, FL: CRC Press, 1997.
20. B. Preneel, R. Govaerts, and J. Vandewalle, "Hash functions based on block ciphers: a synthetic approach," *Advances in Cryptology – CRYPTO '93, Lecture Notes in Computer Science*, vol. 773, D. G. Stinson ed., Springer-Verlag, pp. 368-378, 1994.
21. R. A. Rueppel, "Stream ciphers," *Contemporary Cryptology: The Science of Information Integrity*, G. Simmons ed., pp. 65-134. New York: IEEE Press, 1991.
22. B. Schneier, *Applied Cryptography*. New York: Wiley, 1996.
23. R. Taylor, "An integrity check value algorithm for stream ciphers," *Advances in Cryptology – CRYPTO '93, Lecture Notes in Computer Science*, vol. 773, D. G. Stinson ed., Springer-Verlag, pp. 40-48, 1994.

LILI Keystream Generator

Leonie Ruth Simpson¹, E. Dawson¹, Jovan Dj. Golić², and William L. Millan¹

¹ Information Security Research Centre, Queensland University of Technology,
GPO Box 2434, Brisbane Q 4001, Australia
{simpson,dawson,millan}@fit.qut.edu.au

² Faculty of Electrical Engineering, University of Belgrade,
Bulevar Revolucije 73, 11001 Belgrade, Yugoslavia
golic@galeb.etf.bg.ac.yu

Abstract. A family of keystream generators, called the LILI keystream generators, is proposed for use in stream cipher applications and the security of these generators is investigated with respect to currently known attacks. The design is simple and scalable, based on two binary linear feedback shift registers combined in a simple way, using both irregular clocking and nonlinear functions. The design provides the basic security requirements such as a long period and high linear complexity, and is resistant to known cryptanalytic attacks.

1 Introduction

In this paper, a family of keystream generators based on irregularly clocked LFSRs, intended for use in stream cipher applications, is proposed. We call these the LILI generators. The security of the LILI keystream generators is investigated with respect to currently known attacks on stream ciphers. The keystreams produced are shown to possess the basic security requirements for cryptographic sequences, such as a long period and high linear complexity. It is shown that, provided suitable parameters are selected, the generators are resistant to currently known cryptanalytic attacks. Security implications of parameter selection are discussed.

The LILI family of keystream generators are based on two binary linear feedback shift registers (LFSRs). Many keystream generator designs are based on shift registers, both for the simplicity and speed of LFSR implementation in hardware and for the long period and good statistical properties LFSR sequences possess. To make use of the good keystream properties while avoiding the inherent linear predictability of LFSR sequences, many constructions introduce nonlinearity, by applying a nonlinear function to the outputs of regularly clocked LFSRs or by irregular clocking of the LFSRs [13]. However, keystream generators using regularly clocked LFSRs are susceptible to correlation attacks, including fast correlation attacks, a concept first introduced in [11]. In a fast correlation attack, the initial states of the component shift registers are reconstructed from a known segment of the generator output sequence, without performing a blind search over all possible shift register initial states. As a means of

achieving immunity to these correlation attacks, keystream generators consisting of irregularly clocked LFSRs were proposed. These keystream generators are also susceptible to certain correlation attacks, such as the generalised correlation attack proposed in [6]. However, no fast correlation attacks on these generators have been published.

As correlation attacks have been successful against keystream generators based on either a nonlinear function of regularly clocked LFSR sequences [16,14] or on irregular clocking of LFSRs [6,17], both approaches are combined for the LILI keystream generators. The use of both nonlinear functions and irregular clocking is not novel, having been employed in previous constructions such as ORYX [19] and SOBER [12]. Both ORYX and SOBER are designs for single generators, with fixed size LFSRs and fixed combining functions. In contrast, this proposal is scalable and so describes a family of keystream generators. Also, weaknesses in the design of ORYX resulted in the provision of a very low level of cryptographic security [20]. Some attacks on the SOBER proposal have also been identified [3]. Although the design for the LILI keystream generators described in this paper is conceptually simple, it produces output sequences with provable properties with respect to basic cryptographic security requirements and also provides security against currently known cryptanalytic attacks.

2 Description of LILI Keystream Generators

The LILI keystream generators are simple and fast keystream generators that use two binary LFSRs and two functions to generate a pseudorandom binary keystream sequence, as illustrated in Figure 1. The components of the keystream generator can be grouped into two subsystems based on the functions they perform: clock control and data generation. The LFSR for the clock-control subsystem is regularly clocked. The output of this subsystem is an integer sequence which controls the clocking of the LFSR within the data-generation subsystem. If regularly clocked, the data-generation subsystem is a simple nonlinearly filtered LFSR [13] (nonlinear filter generator). Hence the LILI generator may be viewed as a clock-controlled nonlinear filter generator. Such a system, with the clock control provided by a stop-and-go generator, was examined in [4]. However, the use of stop-and-go clocking produces repetition of the nonlinear filter generator output in the keystream, which may permit attacks. This system is an improvement on that proposal, as stop-and-go clocking is avoided.

The clock-control subsystem of the keystream generator uses a pseudorandom binary sequence produced by a regularly clocked LFSR, $LFSR_c$, of length L_c and a function, f_c , operating on the contents of k stages of $LFSR_c$ to produce a pseudorandom integer sequence, $c = \{c(t)\}_{t=1}^{\infty}$. For practical applications, it is assumed that the feedback polynomial of $LFSR_c$ is primitive and that the initial state of $LFSR_c$ is not the all zero state. Then $LFSR_c$ produces a maximum-length sequence of period $P_c = 2^{L_c} - 1$. At time instant t , the contents of a fixed set of k stages of $LFSR_c$ are input to f_c and the output of f_c is an integer $c(t)$, such that $c(t) \in \{1, 2, \dots, 2^k\}$. The function f_c is a bijective mapping

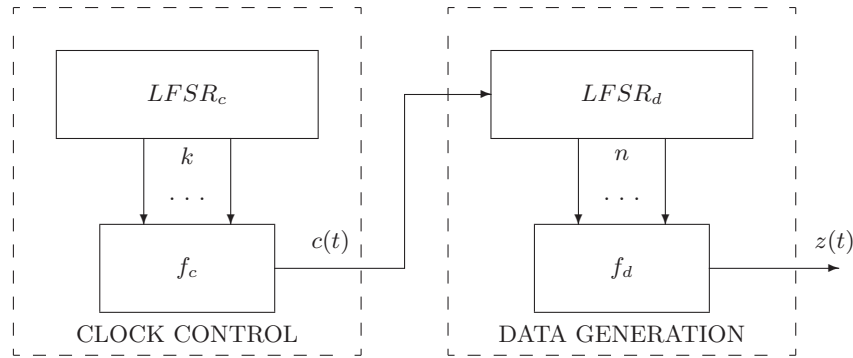


Fig. 1. LILI keystream generator

$\{0, 1\}^k \rightarrow \{1, \dots, 2^k\}$, so that the distribution of integers $c(t)$ is close to uniform. Thus $c = \{c(t)\}_{t=1}^\infty$ is a periodic integer sequence with period equal to P_c . For example, $f_c(x_1, \dots, x_k) = 1 + x_1 + 2x_2 + \dots + 2^{k-1}x_k$ is appropriate.

The variable parameters in the clock-control subsystem are L_c , the feedback function of $LFSR_c$, k , the positions of k stages of $LFSR_c$ used as inputs to the clocking function f_c and f_c itself.

The data-generation subsystem of the keystream generator uses the integer sequence c produced by the clock-control subsystem to control the clocking of a binary LFSR, $LFSR_d$, of length L_d . At time instant t , $LFSR_d$ is clocked $c(t)$ times. The contents of a fixed set of n stages of $LFSR_d$ are input to a Boolean function, f_d . The binary output of f_d forms the keystream bit $z(t)$. After $z(t)$ is produced, $LFSR_c$ is clocked and the process repeated to form the keystream $z = \{z(t)\}_{t=1}^\infty$.

If $LFSR_d$ is regularly clocked, then the data-generation subsystem is simply a nonlinear filter generator. It is assumed that the feedback polynomial of $LFSR_d$ is primitive and that the initial state of $LFSR_d$ is not the all zero state. Then $LFSR_d$ produces a maximum-length sequence of period $P_d = 2^{L_d} - 1$. The output of a regularly clocked nonlinear filter generator is a periodic binary sequence, $g = \{g(i)\}_{i=1}^\infty$, with period dividing P_d . The following basic result is proved in [18].

Theorem 1. *Let $LFSR_d$ have a primitive feedback polynomial and a nonzero initial state. If f_d is balanced, or if P_d is a prime and f_d is not a constant function (zero or one), then the period of g is P_d .*

Now, considering the irregular clocking of $LFSR_d$, the keystream z may be viewed as an irregularly decimated version of the nonlinearly filtered $LFSR_d$ sequence g , with the decimation under the control of $LFSR_c$, so that $z(t) = g(\sum_{j=1}^t c(j))$.

The variable parameters in the data-generation subsystem are L_d , the feedback function of $LFSR_d$, n , the positions of n stages of $LFSR_d$ used as inputs to the filter function f_d and f_d itself. The function f_d should be balanced, highly nonlinear and offer some order of correlation immunity relative to the positions

of n stages used as inputs to f_d (see [9]). The nonlinearity of a Boolean function is defined to be the minimum Hamming distance between the function and any affine function of the same inputs. The correlation-immunity order of a Boolean function is defined to be the maximum nonnegative integer m such that the output is statistically independent of any subset of m inputs, provided that the inputs are uniformly distributed and statistically independent.

3 Keystream Properties

Several properties of pseudorandom binary sequences are considered basic security requirements: a sequence that does not possess these properties is generally considered unsuitable for cryptographic applications. Basic requirements for pseudorandom binary sequences are a long period, high linear complexity and good statistics regarding the distribution of zeroes and ones in the output. High linear complexity avoids an attack using the Berlekamp-Massey [10] algorithm, which requires a length of keystream only twice the linear complexity of the sequence to produce the entire keystream. A bias in the distribution of zeroes and ones in the keystream can be used to reduce the unpredictability of the keystream sequence. These basic requirements are addressed with respect to the LILI family of keystream generators in the remainder of this section.

3.1 Period

The maximum value for the period of z and the conditions under which this value is obtained are given in the following theorem. The result is easily obtained from Theorem 1 and the application of a result regarding the period of irregularly decimated sequences from [2].

Theorem 2. *Let both $LFSR_c$ and $LFSR_d$ have primitive feedback polynomials and nonzero initial states. If $2^{L_d} - 1$ is a prime and f_d is not a constant function or if f_d is balanced and $2^{L_c-1}(2^k + 1) - 1$ is relatively prime to $2^{L_d} - 1$ (provided that $f_c(0, \dots, 0) = 1$), then the period of the output sequence z is given by the product $P_z = (2^{L_c} - 1)(2^{L_d} - 1)$.*

Note that this period implies that each distinct initial state results in the production of a distinct keystream, avoiding the reduction in keyspace which commonly occurs in keystream generators using irregular clocking, where several initial states produce the same keystream [17,12].

3.2 Linear Complexity

For the proposed keystream generator, the output of a nonlinear filter generator with period $P_d = 2^{L_d} - 1$ or a divisor of P_d is nonuniformly decimated by means of a sequence with period $P_c = 2^{L_c} - 1$. In [5], the following upper bound on the linear complexity of irregularly decimated maximum-length sequences is given.

Table 1. Period and linear complexity of binary sequences produced by LILI keystream generators

| $k = 2$ | | | | | $k = 3$ | | | | |
|---------|-------|-------|-------|----------------------------|---------|-------|-------|-------|----------------------------|
| L_c | L_d | P_z | L_z | $\binom{L_d}{2} \cdot P_c$ | L_c | L_d | P_z | L_z | $\binom{L_d}{2} \cdot P_c$ |
| 3 | 4 | 105 | 64 | 42 | 4 | 4 | 225 | 150 | 90 |
| 3 | 6 | 441 | 147 | 105 | 4 | 6 | 945 | 303 | 225 |
| 3 | 7 | 889 | 196 | 147 | 4 | 7 | 1905 | 420 | 315 |
| 3 | 12 | 28665 | 546 | 462 | 4 | 12 | 61425 | 1170 | 990 |
| 7 | 4 | 1905 | 1001 | 762 | 6 | 4 | 945 | 503 | 378 |
| 7 | 6 | 8001 | 2667 | 1905 | 7 | 6 | 8001 | 2373 | 1905 |
| 7 | 7 | 16129 | 3556 | 2667 | 7 | 7 | 16129 | 3556 | 2667 |

When a maximum-length sequence of period P_d is nonuniformly decimated by means of a decimating sequence of period P_c , if the sum modulo P_d of P_c successive values of the decimating sequence equals S , then the decimated sequence has a maximum linear complexity of $L_d \cdot P_c$ only if the multiplicative order of 2 modulo $P_d/\gcd(P_d, S)$ is equal to L_d . Note that this condition is satisfied if $\gcd(P_d, S) = 1$. In [5] it is also shown that if the decimating sequence is randomly chosen, then the probability that maximum linear complexity is obtained can be made arbitrarily close to one for appropriately chosen L_d and P_c .

For a nonuniformly decimated nonlinearly filtered LFSR sequence, the maximal attainable linear complexity is $L'_d \cdot P_c$, where L'_d is the linear complexity of the (regularly clocked) nonlinearly filtered $LFSR_d$ sequence. It is known (e.g., see [13]) that L'_d depends on the filter function and on the positions of stages used for its inputs and that L'_d is very likely to be lower bounded by $\binom{L_d}{r}$, where r is the nonlinear algebraic order of the filter function. Accordingly, our conjecture is that the linear complexity of a nonuniformly decimated nonlinearly filtered $LFSR_d$ sequence is very likely to be lower-bounded by $\binom{L_d}{r} \cdot P_c$. As a consequence, it is also lower-bounded by $L_d \cdot P_c$.

To investigate this conjecture, computer simulations were performed for keystream generators as described in Section 2, for various small shift register lengths. In each case, a nonlinear 3-input balanced nonlinear Boolean function, with $r = 2$, was used as a nonlinear combining function, and the stages of $LFSR_d$ used for inputs to the filter function were selected to form a full positive difference set. That is, the distances between any two stages are distinct. For each keystream generator, a keystream sequence of length greater than the maximum period of the keystream was produced and the period, P_z , and linear complexity, L_z , of the sequence were determined. These values are recorded in Table 1, and support both the theorem regarding the period and the conjecture regarding the linear complexity.

3.3 Statistical Properties of Output Sequence

Under regular clocking, one period of the sequence d produced by $LFSR_d$ when regularly clocked contains $2^{L_d-1} - 1$ zeroes and 2^{L_d-1} ones. For a balanced filter

function such that $f_d(0, \dots, 0) = 0$, a segment of length $2^{L_d} - 1$ of the regularly clocked nonlinear filter generator output sequence g has the same distribution of zeroes and ones as d . When the clocking of $LFSR_d$ is under the control of $LFSR_c$ and when the period of z is $(2^{L_c} - 1)(2^{L_d} - 1)$, then each pair of $LFSR_c$ and $LFSR_d$ states occurs exactly once in a period of z . Therefore one period of z contains $(2^{L_c} - 1)(2^{L_d-1} - 1)$ zeroes and $(2^{L_c} - 1)2^{L_d-1}$ ones, thus maintaining the same proportion of zeroes and ones as in d .

At a more detailed level, the choice of filter function has an effect on the keystream statistics. For a regularly clocked nonlinear filter generator the output sequence may not possess good statistics as the inputs to the filter function are correlated rather than independent. To guarantee good statistical properties, the nonlinear filter function can be chosen to be linear in either the first or the last variable [9].

3.4 Throughput Rate

In producing the keystream, $LFSR_d$ is clocked $c(t)$ times before $z(t)$ is produced. Thus $LFSR_d$ is clocked at least once and at most 2^k times before each keystream bit is produced, with the distribution of values of $c(t)$ almost uniform. Over one period of c , $LFSR_d$ is clocked $\sum_{t=1}^{P_c} c(t) = 2^{L_c-1}(2^k + 1) - 1$ times so, on average, $LFSR_d$ is clocked $\frac{2^{L_c-1}(2^k+1)-1}{2^{L_c}-1}$ times per keystream symbol produced. For large L_c , this is approximately $\frac{2^k+1}{2}$. Thus, for large L_c , the throughput rate is approximately $\frac{2}{2^k+1}$ of the rate at which $LFSR_d$ is clocked, provided an appropriate buffer is used. If not, then one must allow 2^k clocks of $LFSR_d$ per each keystream bit. However, the use of a buffer is very sensitive in high-speed applications.

Alternatively, to achieve the the maximum throughput rate of 1, instead of irregularly clocking the shift register a given number of steps, multiple copies of the feedback function can be maintained, one for each possible value of $c(t)$. The irregular clocking can then be performed in one step only (both in hardware and software). Thus there is a tradeoff between hardware space and timing regularity. Note that the use of either a buffer or parallel-feedback method would provide resistance against timing attacks.

4 Possible Attacks

A number of attacks should be considered with respect to the LILI family of keystream generators. These are known-plaintext attacks conducted under the assumption that the cryptanalyst knows the complete structure of the generator, and the secret key is only the initial states of the component shift registers. For all attacks, the given keystream is viewed as an irregularly decimated version of a nonlinearly filtered $LFSR_d$ sequence, with the decimation under the control of $LFSR_c$. For keystream generators based on more than one LFSR where the key consists of the initial states of the LFSRs, such as the LILI generators,

divide-and-conquer attacks on individual LFSRs should be considered. We deal firstly with divide-and-conquer attacks that target $LFSR_d$, and then with those attacks that target $LFSR_c$.

4.1 Attacks on Irregularly Clocked $LFSR_d$

Suppose a keystream segment of length N is known, say $\{z(t)\}_{t=1}^N$. This is a decimated version of a segment of length M of the underlying regularly clocked nonlinearly filtered $LFSR_d$ sequence, $g = \{g(i)\}_{i=1}^M$, where $M \geq N$. The objective of correlation attacks targeting $LFSR_d$ is to recover the initial state of $LFSR_d$ by identifying the segment $\{g(i)\}_{i=1}^M$ that $\{z(t)\}_{t=1}^N$ was obtained from through decimation, using the correlation between the regularly clocked sequence and the keystream, without knowing the decimating sequence.

For clock-controlled shift registers with constrained clocking, correlation attacks based on a constrained Levenshtein distance and on a probabilistic measure of correlation are proposed in [6] and [7], respectively, and further analysed in [8]. These attacks could be adapted to be used as the first stage of a divide-and-conquer attack on the LILI keystream generators.

For a candidate initial state of $LFSR_d$, say $\{\hat{d}(i)\}_{i=1}^{L_d}$, use the known $LFSR_d$ feedback function to generate a segment of the $LFSR_d$ sequence, $\{\hat{d}(i)\}_{i=1}^{M+L_d-1}$, for some $M \geq L_d$. Then use the known filter function f_d to generate a segment of length M of the output of the nonlinear filter generator when regularly clocked, $\{\hat{g}(i)\}_{i=1}^M$. A measure of correlation between $\{\hat{g}(i)\}_{i=1}^M$ and $\{z(t)\}_{t=1}^N$ is calculated, (either the Constrained Levenshtein Distance (CLD) [6], or the Probabilistic Constrained Edit Distance (PCED) [7]) and the process repeated for all $LFSR_d$ initial states.

In either case, the attack is considered successful if only a few initial states are identified. As the correlation attack based on the PCED takes into account the probability distribution of the decimating sequence, it is statistically optimal and may be successful in cases where the embedding attack based on the CLD is not, such as for larger values of k . The value of M is a function of N and k . If $M = 2^k \times N$, then the probability of not identifying the correct $LFSR_d$ initial state is zero.

The second stage of a divide-and-conquer attack on the generator is the recovery of the initial state of the second shift register. This can be performed as in [17]. From the calculation of the edit distance (either CLD or PCED) between $\{\hat{g}(i)\}_{i=1}^M$ and $\{z(t)\}_{t=1}^N$, form the edit distance matrix, and use this to find possible edit sequences. From each possible edit sequence, form a candidate integer sequence $\{\hat{c}(t)\}_{t=1}^N$. From this, the underlying binary sequence $\{\hat{a}(t)\}_{t=1}^N$ and hence the candidate initial state of $LFSR_c$ can be recovered. To determine whether the correct initial states of both LFSRs have been recovered, use both candidate initial states to generate a candidate keystream and compare it with the known keystream segment.

To conduct either of these correlation attacks requires exhaustive search of $LFSR_d$ initial states. For each $LFSR_d$ initial state, the attacks require calculation of either the CLD or the PCED, with computational complexity

$O(N(M - N))$. Finally, further computational complexity is added in finding the corresponding $LFSR_c$ initial state. For either correlation attack, the minimum length of keystream required for a successful attack on $LFSR_d$ is linear in L_d , but exponential or even superexponential in 2^k (see [8]). For $k = 1$, the required keystream length [22] is reasonably small, but a small increase in k will render this length prohibitively large.

4.2 Attacks Targeting $LFSR_c$

A possible approach to attacking the proposed generator is by targeting the clock-control sequence produced by $LFSR_c$. Guess an initial state of $LFSR_c$, say $\{\hat{a}(t)\}_{t=1}^{L_c}$. Use the known $LFSR_c$ feedback function and the function f_c to generate the decimating sequence $\{\hat{c}(t)\}_{t=1}^N$ for some $N \geq L_c$. Then position the known keystream bits $\{z(t)\}_{t=1}^N$ in the corresponding positions of $\{\hat{g}(i)\}_{i=1}^\infty$, the nonlinear filter generator output when regularly clocked. At this point we have some (not all consecutive) terms in the nonlinear filter generator output sequence and are trying to reconstruct a candidate initial state for $LFSR_d$. The attack could then proceed in several ways.

Consistency Attack. One method is to use the known filter function f_d to write equations relating terms in the underlying $LFSR_d$ sequence to terms in $\{\hat{g}(i)\}_{i=1}^\infty$. Reject the guessed initial state $\{\hat{c}(t)\}_{t=1}^{L_c}$ when the equations are inconsistent. This is a generalisation of the linear consistency test [21]. The feasibility of such an approach depends on the number, n , of inputs to f_d , on the tap positions producing these inputs and on some properties of f_d such as its nonlinearity and order of correlation immunity.

Attacks on Regularly Clocked $LFSR_d$. An alternative approach would be to use a correlation attack on the nonlinear filter generator [14] to recover a linear transform of the $LFSR_d$ sequence, and then recover the $LFSR_d$ initial state. However, this is complicated by not having consecutive terms in the regularly clocked nonlinear filter generator sequence. The feasibility of such an attack primarily depends on the use of a feedback polynomial of $LFSR_d$ that is of low weight or has low-weight polynomial multiples and on the nonlinearity of f_d .

An alternative correlation attack on a (regularly clocked) nonlinear filter generator which could be applied at this point is the conditional correlation attack [1], with a difference that the known output bits are not consecutive. The feasibility of such an attack depends on n and on the tap positions. The use of a full positive difference set for the tap positions, as suggested in [9], and of filter functions with correlation-immunity order greater than zero would render this attack infeasible.

Finally, the inversion attack [9] can be adapted to deal with the case of non-consecutive output bits, but the associated branching process is then supercritical, because more than one bits have to be guessed at a time. As a consequence, the computational complexity may be prohibitively high even if the tap positions are not spread across the $LFSR_d$ length.

Applying any of these approaches requires exhaustive search over the $LFSR_c$ initial state space and additional computation for each candidate $LFSR_c$ state. However, as only some (not all consecutive) terms in the nonlinear filter generator output sequence are available, the required additional computation appears to be prohibitive, especially for highly nonlinear filter functions with a large number of inputs and sufficiently high correlation-immunity order, for the tap positions chosen according to a full positive difference set and for the feedback polynomial of $LFSR_d$ not having low-weight polynomial multiples of relatively small degrees.

5 Choice of Parameters

As an initial security consideration, we should choose the sizes of the shift registers so that exhaustive search of the initial states is prohibitive; at present we recommend that $L_c + L_d > 100$. For a keysize in line with the AES specifications for block ciphers, use $L_c + L_d = 128$. To prevent divide-and-conquer attacks, neither L_d nor L_c should be small. To ensure a large period and good statistical properties, the feedback polynomials of both $LFSR_c$ and $LFSR_d$ should be primitive. In addition, as noted in Section 3, for generator parameters satisfying the conditions of Theorem 2, the period of the output sequence z attains a maximum value of $(2^{L_c} - 1)(2^{L_d} - 1)$, implying that every initial state generates a distinct keystream. Furthermore, the selection of parameters should reduce the possibility of the attacks discussed in Section 4. We address each subsystem of the keystream generators in turn.

5.1 Clock Control

The number, k , of taps from $LFSR_c$ used to form the clocking sequence c affects the period of the output sequence and the resistance against the correlation attacks on irregularly clocked $LFSR_d$, described in Section 4.1, and is the sole factor determining the output rate of the generator. To this end, we recommend $k > 1$ (e.g., $k = 2$ or $k = 3$). The choice of the tap positions does not seem to be important with respect to known attacks, but to be on the safe side, we recommend the use of full positive difference sets.

Also, if the conditions of Theorem 2 are not satisfied, then the period of z is upper-bounded by the product of the period of c and any factors of the period of the nonlinear filter generator output (if regularly clocked) which are relatively prime to $2^{L_c-1}(2^k + 1) - 1$. Thus, for any chosen value of k , $\gcd(2^{L_c-1}(2^k + 1) - 1, 2^{L_d} - 1)$ should be calculated, and the keystream period is maximised when this is one.

5.2 Data Generation

Firstly, the feedback polynomial of $LFSR_d$ should not have low-weight polynomial multiples of relatively small degrees, in order to avoid the vulnerability to fast correlation attacks on $LFSR_d$ when regularly clocked.

Secondly, the number, n , and positions of taps for the filter function, f_d , should be chosen so as to ensure the resistance to attacks discussed in Section 4.2. For example, we recommend that $n \geq 10$ and that the tap positions form a full positive difference set if possible.

Thirdly, the filter function, f_d , should be balanced in order to achieve good statistical properties and a large period (Theorem 1).

Fourthly, f_d should be chosen so as to reduce the possibility of attacks discussed in Section 4.2 (especially if $k = 1$). To this end, f_d should have high correlation-immunity order and high nonlinearity. The proportion of balanced Boolean functions which offer any nonzero order of correlation immunity is small, making it unlikely that a randomly generated function will meet these criteria. Instead, a filter function should be constructed to obtain the required properties. Since there are tradeoffs between nonlinearity, correlation-immunity order and algebraic order, we seek functions that optimise these bounds.

In [15], it was proven that balanced Boolean functions exist with 10 inputs, correlation-immunity order 3, algebraic order 6 and nonlinearity 480. In the same paper a function with CI(1), algebraic order 8 and nonlinearity 484 was constructed. Both of these Boolean functions maximise the Siegenthaler tradeoff and they have the highest possible nonlinearity for their given order of correlation immunity, so either would be a good choice for the output function f_d . For our example, we choose a CI(3) function as we believe that gives a greater resistance to conditional correlation attacks.

6 Example

For a 128-bit key, we select the lengths of $LFSR_c$ and $LSFR_d$ to be 39 and 89, respectively. The feedback polynomials of both $LFSR_c$ and $LFSR_d$ are the primitive polynomials of degrees 39 and 89, respectively, listed in the Appendix.

For the clock-control subsystem, the length of $LFSR_c$ is $L_c = 39$, from which $k = 2$ bits are selected to determine the number of data clocks by the natural mapping: $f_c(x_1, x_2) = 1 + x_1 + 2x_2$.

For the data-generation subsystem, we let $n = 10$. Now, we have $L_d \geq 80$ and this permits the positions of inputs to f_d to form a full positive difference set, shown in the Appendix. Also, we select f_d from [15] to be a balanced, CI(3) function of 10 inputs, with nonlinearity 480 and algebraic order $r = 6$ (see the Appendix for the truth table).

6.1 Properties

As the feedback polynomial of $LFSR_d$ is primitive, f_d is balanced and in addition $2^{89} - 1$ is a Mersenne prime, the conditions of Theorem 2 are satisfied. Thus the period of the keystream is $P_z = (2^{39} - 1)(2^{89} - 1)$. According to Section 3.2, the linear complexity of the keystream sequence is conjectured to be at least $\binom{L_d}{r} \cdot P_c = \binom{89}{6} \cdot (2^{39} - 1) \approx 2^{68}$. With regard to the security offered by this value, we note that this means that about 2^{69} known plaintext bits must be

intercepted in order to perform the Berlekamp-Massey [10] attack. As the key will be changed well before even a fraction of this amount of data is generated, LILI is considered to be secure from such an attack.

6.2 Possible Attacks

Both the period and the conjectured linear complexity of the keystream are too large to be used in cryptanalytic attacks.

The choice of parameters for the data-generation subsystem, in particular the Boolean function f_d , make attacks targetting $LFSR_c$, outlined in Section 4.2, infeasible. In [14], fast correlation attacks on regularly clocked nonlinear filter generators with low-weight feedback polynomials and a known keystream segment of 20,000 bits were not successful when the probability of noise, p , exceeded 0.45. The computational complexity of these attacks is proportional to the length of keystream used and the average number of parity checks used per keystream bit. For the assumed function f_d , the probability of noise is given as $p = 0.46875$, so that the amount of keystream required would be much greater than 20,000 bits. This is likely to make the complexity of an attack on a regularly clocked nonlinear filter generator prohibitive, even if enough low-weight polynomial multiples of the $LFSR_d$ feedback polynomial, used to form parity checks, could be obtained. Given that the keystream segment is from a clock-controlled nonlinear filter generator and that the $LFSR_d$ feedback polynomial does not have low-weight polynomial multiples of relatively small degrees, such an attack appears infeasible.

The length of $LFSR_d$ makes attacks targetting $LFSR_d$, outlined in Section 4.1, infeasible as these attacks require exhaustive search of the initial states of $LFSR_d$, performing some calculation of the correlation for each state. The complexity of such attacks is $O((2^{89} - 1)(3N^2))$, where the required length of the known keystream, N , is very likely to be very large even for $k = 2$. In [17], successful probabilistic correlation attacks were performed on the shrinking generator for given keystream lengths of twenty times the length of the underlying LFSR. The deletion rate for this example is similar, so an estimate of the complexity of these attacks is $O(2^{112})$.

7 Conclusion

In this paper, a family of keystream generators, intended for use in stream cipher applications, is proposed. The design is both simple and scalable: the generators are based on two binary LFSRs and use two combining functions. The security of these keystream generators is investigated. For appropriately chosen components, the generators are shown to provide the basic security requirements for cryptographic sequences, such as a long period and high linear complexity. Also, they are immune to current known-plaintext attacks, conducted under the assumption that the cryptanalyst knows the entire structure of the generator and the secret key is only the initial states of the two LFSRs.

To select an instance from the proposed family, it is necessary to select appropriate values for L_c , L_d , k and n , to have primitive feedback polynomials of both LFSRs and a highly nonlinear balanced Boolean function of an appropriate correlation-immunity order for the filter function. The selection of components can maximise the period and minimise the chances of a successful cryptanalytic attack. The use of both nonlinear combining functions and irregular clocking in LFSR based stream ciphers is not a novel proposal, and has been employed in previous constructions. However, in this proposal the two approaches are combined in a manner that produces output sequences with provable properties with respect to basic cryptographic security requirements and also provides security against currently known cryptanalytic attacks.

Appendix

Full Details of Example LILI with 128 Bit Key

The LFSRs have these feedback polynomials:

$$\begin{aligned} LFSR_c &: x^{39} + x^{35} + x^{33} + x^{31} + x^{17} + x^{15} + x^{14} + x^2 + 1 \\ LFSR_d &: x^{89} + x^{83} + x^{80} + x^{55} + x^{53} + x^{42} + x^{39} + x + 1. \end{aligned}$$

The two inputs x_1, x_2 to f_c are taken from $LFSR_c$ positions 12 and 20, where the range is $[0, 38]$.

The 10 inputs to f_d are taken from $LFSR_d$ positions according to this full positive difference set: (0, 1, 3, 7, 12, 20, 30, 44, 65, 80).

The truth table of the output function f_d :

```

0,0,1,1,1,1,0,0,1,1,0,0,0,0,1,1,1,1,0,0,0,0,1,1,0,0,1,1,1,1,0,0,
0,0,1,1,1,1,0,0,1,1,0,0,0,0,1,1,1,1,0,0,0,0,1,1,0,0,1,1,1,1,0,0,
0,0,1,1,1,1,0,0,1,1,0,0,0,0,1,1,1,1,0,0,0,0,1,1,0,0,1,1,1,1,0,0,
1,1,0,0,0,0,1,1,0,0,1,1,1,1,0,0,0,0,1,1,1,1,0,0,1,1,0,0,0,0,1,1,
0,1,0,1,1,0,1,0,1,0,1,0,0,1,0,1,1,0,1,0,0,1,0,1,0,1,0,1,0,1,0,1,0,
0,1,0,1,1,0,1,0,1,0,1,0,0,1,0,1,1,0,1,0,0,1,0,1,0,1,0,1,0,1,0,1,0,
0,1,0,1,1,0,1,0,1,0,1,0,0,1,0,1,1,0,1,0,0,1,0,1,0,1,0,1,0,1,0,1,0,
1,0,1,0,0,1,0,1,0,1,0,1,0,1,1,0,1,0,0,1,0,1,1,0,1,0,1,0,1,0,1,0,1,
0,1,1,0,0,1,1,0,1,0,0,1,1,0,0,1,1,0,0,1,1,0,0,1,0,1,1,0,0,1,1,0,
0,1,1,0,0,1,1,0,1,0,0,1,1,0,0,1,1,0,0,1,1,0,0,1,0,1,1,0,0,1,1,0,
0,1,1,0,1,0,0,1,0,1,1,0,1,0,0,1,1,0,0,1,0,1,1,0,1,0,0,1,0,1,1,0,
0,1,1,0,1,0,0,1,1,0,0,1,0,1,1,0,0,1,1,0,0,1,1,0,0,1,0,1,1,0,
0,0,0,0,1,1,1,1,1,1,1,1,0,0,0,0,1,1,1,1,0,0,0,0,0,0,0,0,1,1,1,1,
1,1,1,1,0,0,0,0,0,0,0,0,1,1,1,1,0,0,0,0,1,1,1,1,1,1,1,1,1,0,0,0,0,
0,0,1,1,0,0,1,1,1,1,0,0,1,1,0,0,1,1,0,0,1,1,0,0,0,0,1,1,0,0,1,1,
1,1,0,0,1,1,0,0,0,0,1,1,0,0,1,1,0,0,1,1,1,1,0,0,1,1,1,0,0,1,1,0,0,
0,0,1,1,1,1,0,0,0,0,1,1,1,1,0,0,1,1,0,0,0,0,1,1,1,1,0,0,0,0,1,1,
1,1,0,0,0,0,1,1,1,1,0,0,0,0,1,1,0,0,1,1,1,1,0,0,0,0,1,1,1,1,0,0,
0,0,1,1,1,1,0,0,1,1,0,0,0,0,1,1,0,0,1,1,1,1,0,0,1,1,0,0,0,0,1,1,

```

1,1,0,0,0,0,1,1,0,0,1,1,1,0,0,1,1,0,0,0,0,1,1,0,0,1,1,1,0,0,
0,1,0,1,0,1,0,1,1,0,1,0,1,0,1,0,1,0,1,0,1,0,0,1,0,1,0,1,0,1,
1,0,1,0,1,0,1,0,0,1,0,1,0,1,0,1,0,1,0,1,0,1,1,0,1,0,1,0,1,0,
0,1,0,1,1,0,1,0,0,1,0,1,1,0,1,0,1,0,1,0,0,1,0,1,1,0,1,0,0,1,0,1,
1,0,1,0,0,1,0,1,1,0,1,0,0,1,0,1,0,1,1,0,1,0,0,1,0,1,1,0,1,0,
0,1,0,1,1,0,1,0,1,0,1,0,0,1,0,1,0,1,1,0,1,0,1,0,1,0,0,1,0,1,
1,0,1,0,0,1,0,1,0,1,0,1,0,1,0,1,0,1,1,0,1,0,0,1,0,1,1,0,1,0,
0,1,0,1,1,0,1,0,1,0,1,0,0,1,0,1,0,1,1,0,1,0,1,0,1,0,0,1,0,1,
1,0,1,0,0,1,0,1,0,1,0,1,1,0,1,0,1,0,1,0,0,1,0,1,0,1,1,0,1,0,
0,1,1,0,0,1,1,0,0,1,1,0,0,1,1,0,1,0,0,1,1,0,0,1,1,0,0,1,1,0,
0,1,1,0,0,1,1,0,0,1,1,0,0,1,1,0,1,0,0,1,1,0,0,1,1,0,0,1,1,0,
1,0,0,1,1,0,0,1,1,0,0,1,1,0,0,1,0,1,1,0,0,1,1,0,0,1,1,0,1,0,
0,1,1,0,0,1,1,0,1,0,0,1,1,0,0,1,0,1,1,0,0,1,1,0,1,0,0,1,1,0,
1,0,0,1,1,0,0,1,0,1,1,0,0,1,1,0,1,0,0,1,1,0,0,1,1,0,0,1,1,0,
0,1,1,0,0,1,1,0,1,0,0,1,1,0,0,1,0,1,1,0,0,1,1,0,1,0,0,1,1,0,
1,0,0,1,1,0,0,1,0,1,1,0,0,1,1,0,1,0,0,1,1,0,0,1,1,0,0,1,1,0,
0,1,1,0,1,0,0,1,0,1,1,0,1,0,0,1,0,1,1,0,1,0,0,1,0,1,1,0,0,1,
1,0,0,1,0,1,1,0,1,0,0,1,0,1,1,0,1,0,0,1,0,1,1,0,1,0,0,1,1,0.

This Boolean Function has 10 inputs and these properties: balanced, CI(3), algebraic order 6, nonlinearity 480, no linear structures.

References

1. R. Anderson. Searching for the Optimum Correlation Attack. In *Fast Software Encryption - Leuven '94*, volume 1008 of *Lecture Notes in Computer Science*, pages 137–143. Springer–Verlag, 1995.
2. G. R. Blakley and G. B. Purdy. A Necessary and Sufficient Condition for Fundamental Periods of Cascade Machines to be Products of the Fundamental Periods of their Constituent Finite State Machines. *Information Sciences*, vol. 24, pp. 71–91, 1981.
3. D. Bleichenbacher and S. Patel. SOBER Cryptanalysis. In *Fast Software Encryption - Rome '99*, volume 1636 of *Lecture Notes in Computer Science*, pages 305–316. Springer–Verlag, 1999.
4. C. Ding, G. Xiao and W. Shan. *The Stability Theory of Stream Ciphers*. Volume 561 of *Lecture Notes in Computer Science*. Springer–Verlag, 1991.
5. J. Dj. Golić and M. Živković. On the Linear Complexity of Nonuniformly Decimated PN-Sequences. *IEEE Trans. Inform. Theory*, vol. IT-34, pp. 1077–1079, 1988.
6. J. Dj. Golić and M. J. Mihaljević. A Generalized Correlation Attack on a Class of Stream Ciphers Based on the Levenshtein Distance. *Journal of Cryptology*, vol. 3(3), pp. 201–212, 1991.
7. J. Dj. Golić and S. Petrović. A Generalized Correlation Attack with a Probabilistic Constrained Edit Distance. In *Advances in Cryptology - EUROCRYPT '92*, volume 658 of *Lecture Notes in Computer Science*, pages 472–476. Springer–Verlag, 1992.
8. J. Dj. Golić and L. O'Connor. Embedding and Probabilistic Correlation Attacks on Clock-Controlled Shift Registers. In *Advances in Cryptology - EUROCRYPT '94*, volume 950 of *Lecture Notes in Computer Science*, pages 230–243. Springer–Verlag, 1994.
9. J. Dj. Golić. On the Security of Nonlinear Filter Generators. In *Fast Software Encryption - Cambridge '96*, volume 1039 of *Lecture Notes in Computer Science*, pages 173–188. Springer–Verlag, 1996.

10. J. Massey. Shift-Register Synthesis and BCH Decoding. *IEEE Trans. Inform. Theory*, vol. IT-15, pp. 122–127, Jan. 1969.
11. W. Meier and O. Staffelbach. Fast Correlation Attacks on Certain Stream Ciphers. *Journal of Cryptology*, vol. 1(3), pp. 159–167, 1989.
12. G. Rose. A Stream Cipher Based on Linear Feedback over $GF(2^8)$. In *Information Security and Privacy - Brisbane '98*, volume 1438 of *Lecture Notes in Computer Science*, pages 135–146. Springer-Verlag, 1998.
13. R. Rueppel. *Analysis and Design of Stream Ciphers*. Springer-Verlag, Berlin, 1986.
14. M. Salmasizadeh, L. Simpson, J. Dj. Golić and E. Dawson. Fast Correlation Attacks and Multiple Linear Approximations. In *Information Security and Privacy - Nepean '97*, volume 1270 of *Lecture Notes in Computer Science*, pages 228–239. Springer-Verlag, 1997.
15. P. Sarkar and S. Maitra. Nonlinearity Bounds and Constructions of Resilient Boolean Functions. In *Advances in Cryptology - CRYPTO 2000*, volume 1880 of *Lecture Notes in Computer Science*, pages 515–532. Springer-Verlag, 2000.
16. T. Siegenthaler. Decrypting a Class of Stream Ciphers Using Ciphertext Only. *IEEE Trans. Computers*, vol. C-34(1), pp. 81–85, 1985.
17. L. Simpson, J. Dj. Golić and E. Dawson. A Probabilistic Correlation Attack on the Shrinking Generator. In *Information Security and Privacy - Brisbane '98*, volume 1438 of *Lecture Notes in Computer Science*, pages 147–158. Springer-Verlag, 1998.
18. L. Simpson. *Divide and Conquer Attacks on Shift Register Based Stream Ciphers*. PhD thesis, Information Security Research Centre, Queensland University of Technology, Brisbane, Australia, November 1999.
19. TIA TR45.0.A, *Common Cryptographic Algorithms*. Telecommunications Industry Association, Vienna V A., USA, June 1995, Rev B.
20. D. Wagner, L. Simpson, E. Dawson, J. Kelsey, W. Millan and B. Schneier. Cryptanalysis of ORYX. In *Proceedings of the Fifth Annual Workshop on Selected Areas in Cryptography - SAC '98*, volume 1556 of *Lecture Notes in Computer Science*, pages 296–305. Springer-Verlag, 1998.
21. K. C. Zeng, C. H. Yang and T. R. N. Rao. On the Linear Consistency Test (LCT) in Cryptanalysis with Applications. In *Advances in Cryptology - CRYPTO '89*, volume 434 of *Lecture Notes in Computer Science*, pages 164–174. Springer-Verlag, 1990.
22. M. Živković. An Algorithm for the Initial State Reconstruction of the Clock-Controlled Shift Register. *IEEE Trans. Inform. Theory*, vol. IT-37, pp. 1488–1490, Sept. 1991.

Improved Upper Bound on the Nonlinearity of High Order Correlation Immune Functions

Yuliang Zheng¹ and Xian-Mo Zhang²

¹ Monash University, Frankston, Melbourne, VIC 3199, Australia
yuliang.zheng@monash.edu.au, <http://www.netcomp.monash.edu.au/links/>

² The University of Wollongong, Wollongong, NSW 2522, Australia
xianmo@cs.uow.edu.au

Abstract. It has recently been shown that when $m > \frac{1}{2}n - 1$, the nonlinearity N_f of an m th-order correlation immune function f with n variables satisfies the condition of $N_f \leq 2^{n-1} - 2^m$, and that when $m > \frac{1}{2}n - 2$ and f is a balanced function, the nonlinearity satisfies $N_f \leq 2^{n-1} - 2^{m+1}$. In this work we prove that the general inequality, namely $N_f \leq 2^{n-1} - 2^m$, can be improved to $N_f \leq 2^{n-1} - 2^{m+1}$ for $m \geq 0.6n - 0.4$, regardless of the balance of the function. We also show that correlation immune functions achieving the maximum nonlinearity for these functions have close relationships with plateaued functions. The latter have a number of cryptographically desirable properties.

Key words: Correlation Immune Functions, Nonlinearity, Resilient Functions, Plateaued Functions, Stream Ciphers

1 Introduction

Correlation immunity has long been recognized as one of the critical indicators of nonlinear combining functions of shift registers in stream generators (see [12]). A high correlation immunity is generally a very desirable property, in view of various successful correlation attacks against a number of stream ciphers (see for instance [6]).

Another class of cryptanalytic attacks against stream ciphers, called best approximation attacks, were advocated in [4]. Success of these attacks in breaking a stream cipher is made possible by exploiting the low nonlinearity of functions employed by the cipher, and it highlights the significance of nonlinearity in the analysis and design of encryption algorithms.

Recently Sarkar and Maitra [10] have proved that when $m > \frac{1}{2}n - 1$, the nonlinearity N_f of an m th-order correlation immune function f with n variables satisfies the condition of $N_f \leq 2^{n-1} - 2^m$. In addition they have shown that if f is balanced and $m > \frac{1}{2}n - 2$, then the condition becomes $N_f \leq 2^{n-1} - 2^{m+1}$. (See also Section 8 for independent efforts by researchers other than Sarkar and Maitra.)

In this work we focus our attention on the case of $m \geq 0.6n - 0.4$. We show that for such m and n , the nonlinearity of an m th-order correlation immune

function f with n variables must satisfy the condition of $N_f \leq 2^{n-1} - 2^{m+1}$, regardless of the balance of the function. This represents an improvement on the upper bound of $N_f \leq 2^{n-1} - 2^m$.

Plateaued functions are a new class of functions recently introduced in [16]. These functions have a number of properties that are deemed desirable in cryptography. We show that, interestingly, a correlation immune function with the maximum nonlinearity achievable by such a function can be identified with a plateaued function. This provides a new avenue for the analysis and design of cryptographically useful correlation immune functions.

The remaining part of this paper is organized as follows: Section 2 introduces basic definitions on Boolean functions, and Section 3 summarizes some of the important cryptographic criteria for Boolean functions. This will be followed by Section 4 where relevant properties of plateaued functions are discussed. Some useful results on correlation immune functions are introduced in Section 5. These results will then be used in Section 6 where our improved upper bound on the nonlinearity of correlation immune functions is proved. In the same section some relationships between correlation immune functions and plateaued functions are also examined. In Section 7, the new upper bound is demonstrated to be tight for balanced correlation immune functions. Finally the paper is closed by Section 8 where possible directions for future research are pointed out.

2 Boolean Functions

We consider functions from V_n to $GF(2)$ (or simply functions on V_n), where V_n is the vector space of n tuples of elements from $GF(2)$. The *truth table* of a function f on V_n is a $(0, 1)$ -sequence defined by $(f(\alpha_0), f(\alpha_1), \dots, f(\alpha_{2^n-1}))$, and the *sequence* of f is a $(1, -1)$ -sequence defined by $((-1)^{f(\alpha_0)}, (-1)^{f(\alpha_1)}, \dots, (-1)^{f(\alpha_{2^n-1})})$, where $\alpha_0 = (0, \dots, 0, 0)$, $\alpha_1 = (0, \dots, 0, 1)$, \dots , $\alpha_{2^n-1} = (1, \dots, 1, 1)$. The *matrix* of f is a $(1, -1)$ -matrix of order 2^n defined by $M = ((-1)^{f(\alpha_i \oplus \alpha_j)})$ where \oplus denotes the addition in V_n . A function f is said to be *balanced* if its truth table contains an equal number of ones and zeros.

Given two sequences $\tilde{a} = (a_1, \dots, a_m)$ and $\tilde{b} = (b_1, \dots, b_m)$, their *component-wise product* is defined by $\tilde{a} * \tilde{b} = (a_1 b_1, \dots, a_m b_m)$. In particular, if $m = 2^n$ and \tilde{a}, \tilde{b} are the sequences of functions f and g on V_n respectively, then $\tilde{a} * \tilde{b}$ is the sequence of $f \oplus g$ where \oplus denotes the addition in $GF(2)$.

Let $\tilde{a} = (a_1, \dots, a_m)$ and $\tilde{b} = (b_1, \dots, b_m)$ be two sequences or vectors, the *scalar product* of \tilde{a} and \tilde{b} , denoted by $\langle \tilde{a}, \tilde{b} \rangle$, is defined as the sum of the component-wise multiplications. In particular, when \tilde{a} and \tilde{b} are from V_m , $\langle \tilde{a}, \tilde{b} \rangle = a_1 b_1 \oplus \dots \oplus a_m b_m$, where the addition and multiplication are over $GF(2)$, and when \tilde{a} and \tilde{b} are $(1, -1)$ -sequences, $\langle \tilde{a}, \tilde{b} \rangle = \sum_{i=1}^m a_i b_i$, where the addition and multiplication are over the reals.

An *affine* function f on V_n is a function that takes the form of $f(x_1, \dots, x_n) = a_1 x_1 \oplus \dots \oplus a_n x_n \oplus c$, where $a_j, c \in GF(2)$, $j = 1, 2, \dots, n$. Furthermore f is called a *linear* function if $c = 0$.

A $(1, -1)$ -matrix N of order n is called a *Hadamard* matrix if $NN^T = nI_n$, where N^T is the transpose of N and I_n is the identity matrix of order n . A Sylvester-Hadamard matrix of order 2^n , denoted by H_n , is generated by the following recursive relation

$$H_0 = 1, H_n = \begin{bmatrix} H_{n-1} & H_{n-1} \\ H_{n-1} & -H_{n-1} \end{bmatrix}, n = 1, 2, \dots$$

Obviously H_n is symmetric. Let ℓ_i , $0 \leq i \leq 2^n - 1$, be the i row of H_n . It is known that ℓ_i is the sequence of a linear function $\varphi_i(x)$ defined by the scalar product $\varphi_i(x) = \langle \alpha_i, x \rangle$, where α_i is the i th vector in V_n according to the ascending alphabetical order.

The *Hamming weight* of a $(0, 1)$ -sequence ξ , denoted by $HW(\xi)$, is the number of ones in the sequence. Given two functions f and g on V_n , the *Hamming distance* $d(f, g)$ between them is defined as the Hamming weight of the truth table of $f(x) \oplus g(x)$, where $x = (x_1, \dots, x_n)$.

3 Cryptographic Criteria of Boolean Functions

The following criteria for cryptographic Boolean functions are often considered: balance, nonlinearity, propagation criterion, correlation immunity, algebraic degree and non-zero linear structures. In this paper we focus mainly on nonlinearity and correlation immunity.

The so called Parseval's equation (Page 416 [7]) is a useful tool in this work: Let f be a function on V_n and ξ denote the sequence of f . Then $\sum_{i=0}^{2^n-1} \langle \xi, \ell_i \rangle^2 = 2^{2^n}$ where ℓ_i is the i th row of H_n , $i = 0, 1, \dots, 2^n - 1$.

The *nonlinearity* of a function f on V_n , denoted by N_f , is the minimal Hamming distance between f and all affine functions on V_n , i.e., $N_f = \min_{i=1,2,\dots,2^{n+1}} d(f, \psi_i)$ where $\psi_1, \psi_2, \dots, \psi_{2^{n+1}}$ are all the affine functions on V_n . High nonlinearity can be used to resist a linear attack. The following characterization of nonlinearity will be useful (for a proof see for instance [8]).

Lemma 1. *The nonlinearity of f on V_n can be expressed by*

$$N_f = 2^{n-1} - \frac{1}{2} \max\{|\langle \xi, \ell_i \rangle|, 0 \leq i \leq 2^n - 1\}$$

where ξ is the sequence of f and $\ell_0, \dots, \ell_{2^n-1}$ are the rows of H_n , namely, the sequences of linear functions on V_n .

From Lemma 1 and Parseval's equation, it is easy to verify that $N_f \leq 2^{n-1} - 2^{\frac{1}{2}n-1}$ for any function f on V_n . If $N_f = 2^{n-1} - 2^{\frac{1}{2}n-1}$, then f is called a *bent function* [9]. It is known that a bent function on V_n exists only when n is even.

Let f be a function on V_n . For a vector $\alpha \in V_n$, denote by $\xi(\alpha)$ the sequence of $f(x \oplus \alpha)$. Thus $\xi(0)$ is the sequence of f itself and $\xi(0) * \xi(\alpha)$ is the sequence of $f(x) \oplus f(x \oplus \alpha)$. Set $\Delta_f(\alpha) = \langle \xi(0), \xi(\alpha) \rangle$, the scalar product of $\xi(0)$ and $\xi(\alpha)$. $\Delta(\alpha)$ is called the auto-correlation of f with a shift α . We omit the subscript of

$\Delta_f(\alpha)$ if no confusion occurs. Obviously, $\Delta(\alpha) = 0$ if and only if $f(x) \oplus f(x \oplus \alpha)$ is balanced, i.e., f satisfies the propagation criterion with respect to α . In the case that f does not satisfy the propagation criterion with respect to a vector α , it may be desirable for $f(x) \oplus f(x \oplus \alpha)$ to be almost balanced. That is, one may require $|\Delta_f(\alpha)|$ to be a small value.

The concept of correlation immune functions was introduced by Siegenthaler [12]. Xiao and Massey gave an equivalent definition [2,5]: A function f on V_n is called a *math-order correlation immune function* if

$$\sum_{x \in V_n} f(x)(-1)^{\langle \beta, x \rangle} = 0$$

for all $\beta \in V_n$ with $1 \leq HW(\beta) \leq m$, where in the sum, $f(x)$ and $\langle \beta, x \rangle$ are regarded as real-valued functions. From the first equality in Section 4.2 of [2], a correlation immune function can also be equivalently restated as follows: Let f be a function on V_n and let ξ be its sequence. Then f is called a *math-order correlation immune function* if $\langle \xi, \ell \rangle = 0$ for every ℓ , where ℓ is the sequence of a linear function $\varphi(x) = \langle \alpha, x \rangle$ on V_n constrained by $1 \leq HW(\alpha) \leq m$. In fact, $\langle \xi, \ell_i \rangle = 0$, where ℓ_i is the i th row of H_n , if and only if $f(x) \oplus \langle \alpha_i, x \rangle$ is balanced, where α_i is the binary representation of an integer i , $0 \leq i \leq 2^n - 1$. Correlation immune functions are used in the design of running-key generators in stream ciphers to resist a correlation attack and the design of hash functions. Relevant discussions on correlation immune functions, more generally on resilient functions, can be found in [15].

Let f be a function on V_n and ξ denote the sequence of f . We introduce two new notations:

1. Set $\mathfrak{S}_f = \{i \mid \langle \xi, \ell_i \rangle \neq 0, 0 \leq i \leq 2^n - 1\}$ where ℓ_i is the i th row of H_n ,
2. set $\mathfrak{S}_f^* = \{\alpha_i \mid \langle \xi, \ell_{\alpha_i} \rangle \neq 0, 0 \leq i \leq 2^n - 1\}$ where α_i is the binary representation of an integer i , $0 \leq i \leq 2^n - 1$ and ℓ_{α_i} is identified with ℓ_i .

\mathfrak{S}_f^* is essentially the same as \mathfrak{S}_f with the only difference being that its elements are represented by a binary vector in V_n . We will simply write \mathfrak{S}_f as \mathfrak{S} and \mathfrak{S}_f^* as \mathfrak{S}^* when no confusion arises. It is easy to verify that $\#\mathfrak{S}_f$ and $\#\mathfrak{S}_f^*$ are invariant under any nonsingular linear transformation on the variables of the function f . $\#\mathfrak{S}_f$ ($\#\mathfrak{S}_f^*$) together with the distribution of \mathfrak{S}_f (\mathfrak{S}_f^*) determines the correlation immunity and other cryptographic properties of a function.

4 An Overview of Plateaued Functions

The concept of plateaued functions was introduced in [16].

Definition 1. Let f be a function on V_n and ξ denote the sequence of f . If there exists an even number r , $0 \leq r \leq n$, such that $\#\mathfrak{S} = 2^r$ and each $\langle \xi, \ell_j \rangle^2$ takes the value of 2^{n-r} or 0 only, where ℓ_j denotes the j th row of H_n , $j = 0, 1, \dots, 2^n - 1$, then f is called a *rth-order plateaued function* on V_n . f is also simply called a plateaued function on V_n if we ignore the particular order r .

Due to Parseval's equation, the condition that $\#\mathfrak{S} = 2^r$ can be obtained from the condition that "each $\langle \xi, \ell_j \rangle^2$ takes the value of 2^{2n-r} or 0 only, where ℓ_j denotes the j th row of H_n , $j = 0, 1, \dots, 2^n - 1$ ". For the sake of convenience, however, we have mentioned both conditions in the definition of plateaued functions.

Some facts about plateaued functions follow: (1) if f is a r th-order plateaued function, then r must be even, (2) f is an n th-order plateaued function if and only if f is bent, (3) f is a 0th-order plateaued function if and only if f is affine.

All the following results can be found in [16].

Theorem 1. *Let f be a function on V_n and ξ denote the sequence of f . Set $p_M = \max\{|\langle \xi, \ell_j \rangle|, j = 0, 1, \dots, 2^n - 1\}$, where ℓ_j is the j th row of H_n . Then the following statements are equivalent: (i) f is a plateaued function on V_n , (ii) $\sum_{j=0}^{2^n-1} \Delta^2(\alpha_j) = \frac{2^{3n}}{\#\mathfrak{S}}$, (iii) the nonlinearity N_f of f satisfies $N_f = 2^{n-1} - \frac{2^{n-1}}{\sqrt{\#\mathfrak{S}}}$, (iv) $p_M \sqrt{\#\mathfrak{S}} = 2^n$, (v) $N_f = 2^{n-1} - 2^{-\frac{n}{2}-1} \sqrt{\sum_{j=0}^{2^n-1} \Delta^2(\alpha_j)}$.*

Theorem 2. *Let f be a function on V_n and ξ denote the sequence of f . Then*

$$\sum_{j=0}^{2^n-1} \Delta^2(\alpha_j) \geq \frac{2^{3n}}{\#\mathfrak{S}}$$

where the equality holds if and only if f is a plateaued function.

Theorem 3. *Let f be a function on V_n and ξ denote the sequence of f . Then the nonlinearity N_f of f satisfies $N_f \leq 2^{n-1} - \frac{2^{n-1}}{\sqrt{\#\mathfrak{S}}}$, where the equality holds if and only if f is a plateaued function.*

Theorem 4. *Let f be a function on V_n and ξ denote the sequence of f . Then the nonlinearity N_f of f satisfies*

$$N_f \leq 2^{n-1} - 2^{-\frac{n}{2}-1} \sqrt{\sum_{j=0}^{2^n-1} \Delta^2(\alpha_j)}$$

where the equality holds if and only if f is a plateaued function on V_n .

Proposition 1. *Let f be a r th-order plateaued function on V_n . Then the nonlinearity N_f of f satisfies $N_f = 2^{n-1} - 2^{n-\frac{r}{2}-1}$.*

5 Some Useful Results on Correlation Immune Functions

Consider a function f on V_n . Denote by $\xi = (a_0, a_1, \dots, a_{2^n-1})$, where $a_j = \pm 1$, the sequence of f . Obviously

$$(a_0, a_1, \dots, a_{2^n-1})H_n = (\langle \xi, \ell_0 \rangle, \langle \xi, \ell_1 \rangle, \dots, \langle \xi, \ell_{2^n-1} \rangle) \quad (1)$$

where ℓ_i is the i th row of H_n , $i = 0, 1, \dots, 2^n - 1$.

Let p be an integer with $1 \leq p \leq n - 1$. Rewrite (1) as

$$(a_0, a_1, \dots, a_{2^n-1})(H_p \times H_{n-p}) = (\langle \xi, \ell_0 \rangle, \langle \xi, \ell_1 \rangle, \dots, \langle \xi, \ell_{2^n-1} \rangle) \quad (2)$$

where \times is the *Kronecker Product* [14].

Let e_i denote the i th row of H_{n-p} , $i = 0, 1, \dots, 2^{n-p} - 1$. For any fixed j with $0 \leq j \leq 2^{n-p} - 1$, comparing the j th, the $(j+2^{n-p})$ th, \dots , the $(j+(2^p-1)2^{n-p})$ th terms in the two sides of (2), we have

$$(a_0, a_1, \dots, a_{2^n-1})(H_p \times e_j^T) = (\langle \xi, \ell_j \rangle, \langle \xi, \ell_{j+2^{n-p}} \rangle, \dots, \langle \xi, \ell_{j+(2^p-1)2^{n-p}} \rangle)$$

Write $\xi = (\xi_0, \xi_1, \dots, \xi_{2^p-1})$ where each ξ_i is of length 2^{n-p} . Then we have

$$(\langle \xi_0, e_j \rangle, \langle \xi_1, e_j \rangle, \dots, \langle \xi_{2^p-1}, e_j \rangle)H_p = (\langle \xi, \ell_j \rangle, \langle \xi, \ell_{j+2^{n-p}} \rangle, \dots, \langle \xi, \ell_{j+(2^p-1)2^{n-p}} \rangle)$$

Hence

$$\begin{aligned} & 2^p(\langle \xi_0, e_j \rangle, \langle \xi_1, e_j \rangle, \dots, \langle \xi_{2^p-1}, e_j \rangle) \\ &= (\langle \xi, \ell_j \rangle, \langle \xi, \ell_{j+2^{n-p}} \rangle, \dots, \langle \xi, \ell_{j+(2^p-1)2^{n-p}} \rangle)H_p \end{aligned} \quad (3)$$

Based on these discussions, we have the following lemma.

Lemma 2. *Let f be an m th-order correlation immune function on V_n , where $m \leq n - 2$, and ξ be the sequence of f . Then $\langle \xi, \ell_{2^{m+1}-1} \rangle \equiv 0 \pmod{2^{m+2}}$ if and only if $\langle \xi, \ell_0 \rangle \equiv 0 \pmod{2^{m+2}}$ where ℓ_0 is the top row of H_n .*

Proof. Set $W = \{\alpha_0, \alpha_{2^{n-m-1}}, \alpha_{2 \cdot 2^{n-m-1}}, \dots, \alpha_{(2^{m+1}-1)2^{n-m-1}}\}$, where each α_j is the binary representation of an integer j . Note that W is an $(m+1)$ -dimensional linear subspace of V_n .

Write $\xi = (\xi_0, \xi_1, \xi_2, \dots, \xi_{2^{m+1}-1})$, where each ξ_i is of length 2^{n-m-1} . Let $p = m + 1$ and $j = 0$ in (3), we have

$$\begin{aligned} & 2^{m+1}(\langle \xi_0, e_0 \rangle, \langle \xi_1, e_0 \rangle, \dots, \langle \xi_{2^{m+1}-1}, e_0 \rangle) \\ &= (\langle \xi, \ell_0 \rangle, \langle \xi, \ell_{2^{n-m-1}} \rangle, \langle \xi, \ell_{2 \cdot 2^{n-m-1}} \rangle, \dots, \langle \xi, \ell_{(2^{m+1}-1)2^{n-m-1}} \rangle)H_{m+1} \end{aligned} \quad (4)$$

where e_0 denotes the 0th row of H_{n-m-1} , i.e., the all-one sequence of length 2^{n-m-1} .

As $HW(\alpha_{j \cdot 2^{n-m-1}}) \leq m$, we have $\langle \xi, \ell_{j \cdot 2^{n-m-1}} \rangle = 0$, where $j = 1, \dots, 2^{m+1} - 2$. Therefore (4) can be rewritten as

$$\begin{aligned} & 2^{m+1}(\langle \xi_0, e_0 \rangle, \langle \xi_1, e_0 \rangle, \dots, \langle \xi_{2^{m+1}-1}, e_0 \rangle) \\ &= (\langle \xi, \ell_0 \rangle, 0, \dots, 0, \langle \xi, \ell_{(2^{m+1}-1)2^{n-m-1}} \rangle)H_{m+1} \end{aligned} \quad (5)$$

Comparing the rightmost term in the two sides of (5), we have

$$2^{m+1}\langle \xi_{2^{m+1}-1}, e_0 \rangle = \langle \xi, \ell_0 \rangle - \langle \xi, \ell_{(2^{m+1}-1)2^{n-m-1}} \rangle \quad (6)$$

Note that the length of $\xi_{2^{m+1}-1}$ and e_0 is even. Hence $\langle \xi_{2^{m+1}-1}, e_0 \rangle$ must be even. From this it follows that $2^{m+1}\langle \xi_{2^{m+1}-1}, e_0 \rangle \equiv 0 \pmod{2^{m+2}}$. Finally, by considering (6), we have proved that $\langle \xi, \ell_{(2^{m+1}-1)2^{n-m-1}} \rangle \equiv 0 \pmod{2^{m+2}}$ if and only if $\langle \xi, \ell_0 \rangle \equiv 0 \pmod{2^{m+2}}$. \square

By choosing a different W in the proof of Lemma 2, we can prove the following lemma in a similar way.

Lemma 3. *Let f be an m th-order correlation immune function on V_n , where $m \leq n - 2$, and ξ be the sequence of f . Let j_0 be an integer satisfying $0 < j_0 \leq 2^n - 1$ and $HW(\alpha_{j_0}) = m + 1$, where α_{j_0} is the binary representation of the integer j_0 . Then $\langle \xi, \ell_{j_0} \rangle \equiv 0 \pmod{2^{m+2}}$ if and only if $\langle \xi, \ell_0 \rangle \equiv 0 \pmod{2^{m+2}}$.*

Lemma 3 allows us to claim

Lemma 4. *Let f be an m th-order correlation immune function on V_n , where $m \leq n - 2$, and ξ be the sequence of f . Let j_0 be an integer satisfying $0 < j_0 \leq 2^n - 1$ and $HW(\alpha_{j_0}) = m + 1$. If $\langle \xi, \ell_{j_0} \rangle \equiv 0 \pmod{2^{m+2}}$ then $\langle \xi, \ell_j \rangle \equiv 0 \pmod{2^{m+2}}$ for any integer j satisfying $HW(\alpha_j) = m + 1$, where α_j is the binary representation of j .*

The condition of $HW(\alpha_j) = m + 1$ in the lemma above can be removed, as is shown below.

Lemma 5. *Let f be an m th-order correlation immune function on V_n , where $m \leq n - 2$, and ξ be the sequence of f . Let j_0 be an integer satisfying $0 < j_0 \leq 2^n - 1$ and $HW(\alpha_{j_0}) = m + 1$, where α_{j_0} is the binary representation of j_0 . If $\langle \xi, \ell_{j_0} \rangle \equiv 0 \pmod{2^{m+2}}$, then $\langle \xi, \ell_i \rangle \equiv 0 \pmod{2^{m+2}}$ for any row ℓ_i of H_n .*

Proof. We use induction on $HW(\alpha_j)$ to prove that $\langle \xi, \ell_j \rangle \equiv 0 \pmod{2^{m+2}}$, where α_j is the binary representation of the subscript j of ℓ_j .

For $0 < HW(\alpha_j) \leq m$, since f is an m th-order correlation immune function, we have $\langle \xi, \ell_j \rangle = 0$. On the other hand, from Lemma 4, we have $\langle \xi, \ell_j \rangle \equiv 0 \pmod{2^{m+2}}$, where ℓ_j is any row of H_n satisfying $HW(\alpha_j) = m + 1$, and α_j is the binary representation of j . Due to Lemma 3, we also have $\langle \xi, \ell_0 \rangle \equiv 0 \pmod{2^{m+2}}$. Hence we have proved $\langle \xi, \ell_j \rangle \equiv 0 \pmod{2^{m+2}}$, when $HW(\alpha_j) \leq m + 1$.

Now assume that $\langle \xi, \ell_j \rangle \equiv 0 \pmod{2^{m+2}}$, when $m + 1 \leq HW(\alpha_j) \leq k \leq n - 1$. Consider the case of $HW(\alpha_j) = k + 1$. Obviously, W can be rewritten as $W = \{\alpha_0, \alpha_{2^{n-k-1}}, \alpha_{2 \cdot 2^{n-k-1}}, \dots, \alpha_{(2^{k+1}-1)2^{n-k-1}}\}$, where each α_j is the binary representation of an integer j . One can see that W is a $(k + 1)$ -dimensional linear subspace.

Let $\xi = (\xi_0, \xi_1, \xi_2, \dots, \xi_{2^{k+1}-1})$, where each ξ_i is of length 2^{n-k-1} . Furthermore, let $p = k + 1$ and $j = 0$ in (3). Then we have

$$\begin{aligned} & 2^{k+1}(\langle \xi_0, e_0 \rangle, \langle \xi_1, e_0 \rangle, \dots, \langle \xi_{2^{k+1}-1}, e_0 \rangle) \\ &= (\langle \xi, \ell_0 \rangle, \langle \xi, \ell_{2^{n-k-1}} \rangle, \langle \xi, \ell_{2 \cdot 2^{n-k-1}} \rangle, \dots, \langle \xi, \ell_{(2^{k+1}-1)2^{n-k-1}} \rangle) H_{k+1} \end{aligned} \quad (7)$$

where e_0 denotes the 0th row of H_{n-k-1} , i.e., the all-one sequence of length 2^{n-k-1} .

By the assumption, we should have $\langle \xi, \ell_j \rangle \equiv 0 \pmod{2^{m+2}}$ where $j = i \cdot 2^{n-k-1}$, $i = 0, 1, \dots, 2^{k+1} - 2$. Note that $k \geq m + 1$. From (7), we have $\langle \xi, \ell_{(2^{k+1}-1)2^{n-k-1}} \rangle \equiv 0 \pmod{2^{m+2}}$. Furthermore, note that HW

$(\alpha_{(2^{k+1}-1)2^{n-k-1}}) = k + 1$. Taking into account Lemma 4, we can conclude that $\langle \xi, \ell_j \rangle \equiv 0 \pmod{2^{m+2}}$, for $HW(\alpha_j) = k + 1$, where α_j is the binary representation of j . This completes the proof. \square

In the following section, we will use these results to improve the upper bound on the nonlinearity of correlation immune functions.

6 Improving Upper Bounds on Nonlinearity

The following lemma will be used in proving Theorem 5.

Lemma 6. *Let f be an m th-order correlation immune function on V_n , where $\frac{1}{2}n - 1 < m \leq n - 2$, and ξ denotes the sequence of f . If $\binom{n}{m+1} > 2^{2n-2m-2}$, then there must be an integer j_0 , $0 \leq j_0 \leq 2^n - 1$, such that $HW(\alpha_{j_0}) = m + 1$ and $\langle \xi, \ell_{j_0} \rangle = 0$, where α_{j_0} is the binary representation of integer j_0 .*

Proof. Since f is an m th-order correlation immune function on V_n , due to Theorem 3 of [10], we have $\langle \xi, \ell \rangle \equiv 0 \pmod{2^{m+1}}$, where ℓ is any row of H_n . Hence $\langle \xi, \ell \rangle \neq 0$ implies that $|\langle \xi, \ell \rangle| \geq 2^{m+1}$. Using Parseval's equation (Page 416 [7]), we have $\#\mathfrak{S} \leq 2^{2n-2m-2}$.

Note that the number of vectors α in V_n , satisfying $HW(\alpha) = m + 1$, is equal to $\binom{n}{m+1} > 2^{2n-2m-2}$. Hence there must be a vector α_{j_0} such that $HW(\alpha_{j_0}) = m + 1$ and $\alpha_{j_0} \notin \mathfrak{S}^*$. As a result, we have $\langle \xi, \ell_{j_0} \rangle = 0$, where α_{j_0} is the binary representation of j_0 . \square

Theorem 5. *Let f be an m th-order correlation immune function on V_n , where $\frac{1}{2}n - 1 < m \leq n - 2$. If $\binom{n}{m+1} > 2^{2n-2m-2}$, then $N_f \leq 2^{n-1} - 2^{m+1}$, where the equality holds if and only if f is a $2(n-m-2)$ th-order plateaued function.*

Proof. By Lemma 6, there must be a vector α_{j_0} such that $HW(\alpha_{j_0}) = m + 1$ and $\langle \xi, \ell_{j_0} \rangle = 0$. Now using Lemma 5, we have

$$\langle \xi, \ell \rangle \equiv 0 \pmod{2^{m+2}} \quad (8)$$

where ℓ is any row of H_n . Lemma 1 implies that $N_f \leq 2^{n-1} - 2^{m+1}$.

Assume that $N_f = 2^{n-1} - 2^{m+1}$. From Lemma 1, we have

$$\max\{|\langle \xi, \ell_i \rangle|, 0 \leq i \leq 2^n - 1\} = 2^{m+2} \quad (9)$$

Combining (8) and (9), we can conclude that $\langle \xi, \ell \rangle = 2^{m+2}$ if $\langle \xi, \ell \rangle \neq 0$. This proves that f is a $2(n-m-2)$ th-order plateaued function.

Conversely, if f is a $2(n-m-2)$ th-order plateaued function, due to Proposition 1, we must have $N_f = 2^{n-1} - 2^{m+1}$. \square

Let n and m be two integers with $n > m > 0$. We claim that the following inequality holds:

$$\binom{n}{m+1} > \left(\frac{n+m+2}{n-m}\right)^{n-m-1} \quad (10)$$

To prove the claim, we set $\rho(i) = \frac{(n-i)(m+2+i)}{(n-m-1-i)(1+i)}$, where $0 \leq i \leq \frac{1}{2}(n-m-2)$. Since $\binom{n}{m+1} = \binom{n}{n-m-1}$, it is easy to verify that

$$\binom{n}{m+1} = \begin{cases} \rho(0)\rho(1)\cdots\rho(\frac{1}{2}(n-m-2)-1)\left(\frac{n+m+2}{n-m}\right), & \text{if } n-m \text{ is even} \\ \rho(0)\rho(1)\cdots\rho(\frac{1}{2}(n-m-3)), & \text{if } n-m \text{ is odd} \end{cases} \quad (11)$$

In addition, one can also verify that ρ satisfies the condition of $\rho(i) < \rho(i-1)$. Hence

$$\binom{n}{m+1} > \begin{cases} (\rho(\frac{1}{2}(n-m-2)))^{\frac{1}{2}(n-m-2)}\left(\frac{n+m+2}{n-m}\right), & \text{if } n-m \text{ is even} \\ (\rho(\frac{1}{2}(n-m-3)))^{\frac{1}{2}(n-m-1)}, & \text{if } n-m \text{ is odd} \end{cases} \quad (12)$$

There exist two cases to be considered: $n-m$ is even and $n-m$ is odd.

In the former case, we note that $\rho(\frac{1}{2}(n-m-2)) = (\frac{n+m+2}{n-m})^2$. Due to (12), we obtain $\binom{n}{m+1} > (\frac{n+m+2}{n-m})^{n-m-1}$.

In the latter case, as $\rho(\frac{1}{2}(n-m-3)) = \frac{(n+m+3)(n+m+1)}{(n-m+1)(n-m-1)} > (\frac{n+m+2}{n-m})^2$, taking into account (12), we have $\binom{n}{m+1} > (\frac{n+m+2}{n-m})^{n-m-1}$. Thus the inequality in (10) is indeed true.

Theorem 6. *Let f be an m th-order correlation immune function on V_n . If m and n satisfy the condition of $0.6n - 0.4 \leq m \leq n - 2$, then $N_f \leq 2^{n-1} - 2^{m+1}$, where the equality holds if and only if f is also a $2(n-m-2)$ th-order plateaued function.*

Proof. One can verify that

$$\frac{n + \lambda_1 + 2}{n - \lambda_1} > \frac{n + \lambda_2 + 2}{n - \lambda_2}$$

for $n > \lambda_1 > \lambda_2 > 0$, where λ_1 and λ_2 are not necessarily integers. Since $m \geq 0.6n - 0.4$, we have

$$\left(\frac{n+m+2}{n-m}\right)^{n-m-1} \geq \left(\frac{n+0.6n-0.4+2}{n-(0.6n-0.4)}\right)^{n-m-1} = 2^{2n-2m-2}.$$

By using (10), we can conclude that $\binom{n}{m+1} > 2^{2n-2m-2}$. Taking into account Theorem 5, we know that the theorem is indeed true. \square

Part (i) of Theorem 4 in [10] states that the nonlinearity N_f of an m th-order correlation immune function f on V_n satisfies $N_f \leq 2^{n-1} - 2^m$, when $m > \frac{1}{2}n - 1$. Our Theorem 6 represents an improvement on the result in [10], especially for the case of $m \geq 0.6n - 0.4$.

As a consequence of Theorem 5 or Theorem 6, a correlation immune function that achieves the maximum nonlinearity for such a function, also satisfies all the properties of plateaued function, as discussed in Section 4. As a result, by taking into account Theorem 2, we have

Corollary 1. *Let f be an m th-order correlation immune function on V_n . If $0.6n - 0.4 \leq m \leq n - 2$, then $N_f \leq 2^{n-1} - 2^{m+1}$, where the equality holds if and only if f is also a $2(n - m - 2)$ th-order plateaued function or the equality in Theorem 2 holds, i.e., $\sum_{j=0}^{2^{n-1}-1} \Delta^2(\alpha_j) = 2^{n+2m+4}$.*

An (n, m, t) -resilient function is an n -input m -output function or mapping F with the property that it runs through every possible output m -tuple an equal number of times when t arbitrary inputs are fixed and the remaining $n - t$ inputs runs through all the 2^{n-t} input tuples once. The concept was introduced by Chor *et al* in [3] and independently, by Bennett *et al* in [1]. Comparing the definition of resilient functions with that of correlation immune functions, one can see that an $(n, 1, t)$ -resilient function coincides with a balanced t th-order correlation immune function on V_n . In this context, Theorem 1 of [15] is of special interest to practitioners alike, as it shows that each non-zero linear combination of the component functions of an (n, m, t) -resilient function is also a balanced t th-order correlation immune function on V_n , giving rise to $2^m - 1$ distinct, balanced t th-order correlation immune functions in total.

To close this section, we point out a result which follows from Theorem 2 of [10] and Theorem 2 in this paper.

Corollary 2. *Let f be an $(n, 1, m)$ -resilient function, where $\frac{1}{2}n - 2 < m \leq n - 3$. Then the nonlinearity N_f of f satisfies $N_f \leq 2^{n-1} - 2^{m+1}$, where the equality holds if and only if f is also a $2(n - m - 2)$ th-order plateaued function or the equality in Theorem 2 holds, i.e., $\sum_{j=0}^{2^{n-1}-1} \Delta^2(\alpha_j) = 2^{n+2m+4}$.*

7 Tightness of the Upper Bound

As Theorem 6 represents an improved upper bound on the nonlinearity of all the correlation immune functions including both balanced and unbalanced ones, we are further interested in the question as to whether the upper bound is tight or not. It turns out that the question can be answered in an affirmative way for balanced correlation immune functions. The approach we take is to actually demonstrate the existence of m th-order correlation immune, balanced functions on V_n , whose nonlinearity N_f satisfies $N_f = 2^{n-1} - 2^{m+1}$.

We note that [11] is the earliest paper to study the nonlinearity of correlation immune functions. Of particular importance are Theorems 9 and 14 in [11] which happen to be also relevant to the current work. Theorem 9 of

[11] proved the equivalence of two different methods for constructing correlation immune functions, while Theorem 4 in the same paper showed how to obtain highly nonlinear correlation immune functions. Let integers n and m satisfy $m + 2 \geq 2^{n-m-2}$ and $n \geq 16$. For such n and m , there exist 2^{n-m-2} non-zero vectors in V_{m+2} , say $\gamma_0, \gamma_1, \dots, \gamma_{2^{n-m-2}-1}$, such that $HW(\gamma_j) \geq m + 1$, where $j = 0, 1, \dots, 2^{n-m-2} - 1$. Define a mapping P from V_{n-m-2} to V_{m+2} such that $P(V_{n-m-2}) = \{\gamma_0, \gamma_1, \dots, \gamma_{2^{n-m-2}-1}\}$, where $P(V_{n-m-2}) = \{P(\delta) | \delta \in V_{n-m-2}\}$. Based on P , we construct a function f on V_n by $f(x) = f(y, z) = P(y)z^T$ where $x = (y, z)$, $y \in V_{n-m-2}$ and $z \in V_{m+2}$. By using Theorems 9 and 14 of [11], modifying the relevant parameters accordingly, and fixing t to 1, we can construct an $(n, 1, m)$ -resilient (balanced) function f whose nonlinearity N_f reaches the upper bound of $2^{n-1} - 2^{m+1}$.

As a concrete example, let $n = 9$ and $m = 5$. Then $m + 2 \geq 2^{n-m-2}$. Set $\gamma_0 = (1, 1, 1, 1, 1, 1)$, $\gamma_1 = (1, 1, 1, 1, 1, 0)$, $\gamma_2 = (1, 1, 1, 1, 0, 1)$ and $\gamma_3 = (1, 1, 1, 1, 0, 1)$. Then each $\gamma_j \in V_7$ and $HW(\gamma_j) \geq 6$. Define a mapping P from V_2 to V_7 such that $P(0, 0) = \gamma_0$, $P(0, 1) = \gamma_1$, $P(1, 0) = \gamma_2$ and $P(1, 1) = \gamma_3$. Based on P , we construct a function f on V_9 by $f(x) = f(y, z) = P(y)z^T$ where $x = (y, z)$, $y \in V_2$ and $z \in V_7$. Theorems 9 and 14 in [11] tell us that f is a 5th-order correlation immune function on V_9 , and the nonlinearity N_f of f achieves $N_f = 2^8 - 2^6 = 192$, the highest possible value for such a function. Since each γ_j is non-zero, f is balanced. One can verify that the function f takes the form of

$$f(y, z) = y_1 z_6 \oplus y_2 z_7 \oplus y_1 y_2 (z_5 \oplus z_6 \oplus z_7) \\ \oplus z_1 \oplus z_2 \oplus z_3 \oplus z_4 \oplus z_5 \oplus z_6 \oplus z_7$$

where $y = (y_1, y_2)$ and $z = (z_1, z_2, z_3, z_4, z_5, z_6, z_7)$.

The above discussions indicate that the upper bound $(2^{n-1} - 2^{m+1})$ is indeed tight for balanced correlation immune functions. While we have not been able to identify whether the bound is also tight for unbalanced correlation immune functions, its implication would be marginal, due to the fact that unbalanced correlation immune functions have found little use in practice.

To close this section, let us note that in [13], an unbalanced 3rd-order correlation immune function on V_6 whose nonlinearity achieves $2^5 - 2^3 = 24$ is constructed. This particular function does not contradict Theorem 5 or Theorem 6, as the specific parameters $n = 6$ and $m = 3$ satisfy neither $\binom{n}{m+1} > 2^{2n-2m-2}$ nor $m \geq 0.6n - 0.4$.

8 Concluding Remarks

Three separate research groups, Sarkar and Maitra, Tarannikov [13], and Zheng and Zhang, have apparently considered the same question on the upper bound on nonlinearity of correlation immune functions, independently of one another. All three groups submitted their research results to CRYPTO2000, although

only Sarkar and Maitra's got accepted. Our current paper contains essentially the same research results included in our CRYPTO2000 submission, minus those that happened to overlap results in Sarkar and Maitra's CRYPTO2000 paper.

Theorem 6 leaves open as to whether the condition of $0.6n - 0.4 \leq m \leq n - 2$ can be relaxed to $\frac{1}{2}n - 1 < m \leq n - 2$ where $n > 6$. We have recently successfully solved this problem [17].

It would also be interesting, albeit purely from a theoretical point of view, to examine whether the bound $N_f = 2^{n-1} - 2^{m+1}$, where $m \geq 0.6n - 0.4$, is also tight for unbalanced m th-order correlation immune functions, and if it is, how to construct such functions.

Acknowledgment

The second author was supported by a Queen Elizabeth II Fellowship (227 23 1002). Both authors would like to thank Yuriy Tarannikov for pointing out an error in an earlier version, and anonymous referees for SAC2000 for their comments that have helped further improve both the upper bound and the presentation of this paper.

References

1. C. H. Bennett, G. Brassard, and J. M. Robert. Privacy amplification by public discussion. *SIAM J. Computing*, 17:210–229, 1988.
2. P. Camion, C. Carlet, P. Charpin, and N. Sendrier. On correlation-immune functions. In *Advances in Cryptology - CRYPTO'91*, volume 576 of *Lecture Notes in Computer Science*, pages 87–100. Springer-Verlag, Berlin, Heidelberg, New York, 1991.
3. Benny Chor, Oded Goldreich, Johan Håstad, Joel Friedman, Steven Rudich, and Roman Smolensky. The bit extraction problem or t -resilient functions. *IEEE Symposium on Foundations of Computer Science*, 26:396–407, 1985.
4. C. Ding, G. Xiao, and W. Shan. *The Stability Theory of Stream Ciphers*, volume 561 of *Lecture Notes in Computer Science*. Springer-Verlag, Berlin, Heidelberg, New York, 1991.
5. Xiao Guo-Zhen and J. L. Massey. A spectral characterization of correlation-immune combining functions. *IEEE Transactions on Information Theory*, 34(3):569–571, 1988.
6. M. Hermelin and K. Nyberg. Correlation properties of the bluetooth combiner generator. In *The 2nd International Conference on Information Security and Cryptology (ICISC'99), Seoul, Korea*, volume 1787 of *Lecture Notes in Computer Science*, pages 17–29. Springer-Verlag, Berlin, Heidelberg, New York, 1999.
7. F. J. MacWilliams and N. J. A. Sloane. *The Theory of Error-Correcting Codes*. North-Holland, Amsterdam, New York, Oxford, 1978.
8. W. Meier and O. Staffelbach. Nonlinearity criteria for cryptographic functions. In *Advances in Cryptology - EUROCRYPT'89*, volume 434 of *Lecture Notes in Computer Science*, pages 549–562. Springer-Verlag, Berlin, Heidelberg, New York, 1990.

9. O. S. Rothaus. On “bent” functions. *Journal of Combinatorial Theory*, Ser. A, 20:300–305, 1976.
10. P. Sarkar and S. Maitra. Nonlinearity bounds and constructions of resilient boolean functions. In *Advances in Cryptology - CRYPTO2000*, volume 1880 of *Lecture Notes in Computer Science*, pages 515–532. Springer-Verlag, Berlin, Heidelberg, New York, 2000.
11. J. Seberry, X. M. Zhang, and Y. Zheng. On constructions and nonlinearity of correlation immune functions. In *Advances in Cryptology - EUROCRYPT’93*, volume 765 of *Lecture Notes in Computer Science*, pages 181–199. Springer-Verlag, Berlin, Heidelberg, New York, 1994.
12. T. Siegenthaler. Correlation-immunity of nonlinear combining functions for cryptographic applications. *IEEE Transactions on Information Theory*, IT-30 No. 5:776–779, 1984.
13. Yuriy Tarannikov. On resilient boolean functions with maximal possible nonlinearity. (<http://eprint.iacr.org/2000/005/>), 2000.
14. R. Yarlagadda and J. E. Hershey. Analysis and synthesis of bent sequences. *IEE Proceedings (Part E)*, 136:112–123, 1989.
15. X. M. Zhang and Y. Zheng. Cryptographically resilient functions. *IEEE Transactions on Information Theory*, 43(5):1740–1747, 1997.
16. Y. Zheng and X. M. Zhang. Plateaued functions. In *Advances in Cryptology - ICICS’99*, volume 1726 of *Lecture Notes in Computer Science*, pages 284–300. Springer-Verlag, Berlin, Heidelberg, New York, 1999.
17. Y. Zheng and X. M. Zhang. Two new results on correlation immune functions, August 2000. (submitted for publication).

Towards Practical Non-interactive Public Key Cryptosystems Using Non-maximal Imaginary Quadratic Orders (Extended Abstract)

Detlef Hühnlein¹, Michael J. Jacobson, Jr.², and Damian Weber³

¹ Secunet Security Networks AG,
Mergenthalerallee 77-81, D-65760 Eschborn, Germany
huehnlein@secunet.de

² Centre for Applied Cryptographic Research,
University of Waterloo,
Waterloo, Ontario, Canada, N2L 3G1
mjjacobs@cacr.math.uwaterloo.ca

³ Märkische Fachhochschule Iserlohn und Hagen,
Haldener Str. 182, D-58095 Hagen, Germany
weber@mfh-iserlohn.de

Abstract. We present a new non-interactive public key distribution system based on the class group of a non-maximal imaginary quadratic order $Cl(\Delta_p)$. The main advantage of our system over earlier proposals based on $(\mathbb{Z}/n\mathbb{Z})^*$ [19,21] is that embedding id information into group elements in a cyclic subgroup of the class group is easy (straight-forward embedding into prime ideals suffices) and secure, since the entire class group is cyclic with very high probability.

In order to compute discrete logarithms in the class group, the KGC needs to know the prime factorization of $\Delta_p = \Delta_1 p^2$. We present an algorithm for computing discrete logarithms in $Cl(\Delta_p)$ by reducing the problem to computing discrete logarithms in $Cl(\Delta_1)$ and either \mathbb{F}_p^* or $\mathbb{F}_{p^2}^*$. We prove that a similar reduction works for arbitrary non-maximal orders, and that it has polynomial complexity if the factorization of the conductor is known.

Keywords: discrete logarithm, non-maximal imaginary quadratic order, non-interactive cryptography, identity based cryptosystem

1 Introduction

Public-key cryptography is undoubtedly one of the core techniques used to enable authentic, non-repudiable and confidential communication. However, a general problem inherent in public-key systems is that one needs to ensure the authenticity of a given public key. The most common way to solve this problem is to introduce a trusted third party, called a *Certification Authority* (CA), which is-

sues certificates for public keys¹. While this approach is widely used in practice, it would be desirable to have an immediate binding between an identity ID_B and its corresponding public key \mathfrak{b} , which allows one to avoid the tedious verification of certificates. This leads to the notion of *identity based cryptosystems*.

Although the paradigm of identity based cryptography was already introduced by Shamir in 1984 [23], it seems that Maurer and Yacobi [19] were the first to propose a *non-interactive* identity based public key cryptosystem in which Bob's public key \mathfrak{b} can be derived efficiently, solely from his public identity information ID_B , by computing a publicly-known embedding function $\mathfrak{b} = f(ID_B)$. The main idea is to use an (ideally cyclic) group G (generated by \mathfrak{g}) in which exponentiation is not only a one-way-function but a *trapdoor-one-way-function*. The key generation center (KGC), a trusted third party responsible for distributing the private keys, knows the trapdoor information and hence is able to compute discrete logarithms in G . Thus, the KGC computes Bob's private key b such that $\mathfrak{g}^b = \mathfrak{b} = f(ID_B)$. The KGC hands over the secret key b to Bob, who can use this key in a conventional ElGamal- or Diffie-Hellman setup. As soon as all users are equipped with their corresponding secret key, the KGC can destroy the trapdoor-information and may cease to exist.

Maurer and Yacobi's initial proposal was to set up a discrete logarithm based system in $G = (\mathbb{Z}/n\mathbb{Z})^*$, where $n = p_1 \cdots p_r$, p_i prime, such that only the KGC, which knows the factorization of n , is able to compute discrete logarithms in G . However, this approach has a number of drawbacks which render such a scheme impractical [20,18,17].

In this paper, we show that using the class group $Cl(\Delta_p)$ of a non-maximal imaginary quadratic order is much better suited for this purpose. As in the original scheme, the KGC knows trapdoor information (the prime factorization of Δ_p) which enables it to compute discrete logarithms, while for anybody else the discrete logarithm problem (DLP) is assumed to be intractable. We generalize the recent result from [12], valid for the very special case of totally non-maximal orders with prime discriminant, to arbitrary non-maximal imaginary quadratic orders. The resulting algorithm reduces the problem of discrete logarithm computation in the class group of a non-maximal order to computing discrete logarithms in the much smaller class group of the corresponding maximal order and a small number of finite fields. Only the KGC, which knows the factorization of Δ_p , can perform this reduction.

As noted above there are a few advantages to our approach. Unlike the case of $(\mathbb{Z}/n\mathbb{Z})^*$, it is heuristically easy to find class groups $Cl(\Delta_p)$ which are *cyclic*, and hence the embedding of an identity ID_B into a group element \mathfrak{b} , for which the discrete logarithm exists, is straightforward. As the results from [20,18] demonstrate, it seems to be no trivial task to find an embedding into a subgroup of $(\mathbb{Z}/n\mathbb{Z})^*$ which does not facilitate factoring n . In fact, the only secure embedding method for $(\mathbb{Z}/n\mathbb{Z})^*$ seems to restrict n to having *only two* large prime factors p_1

¹ We assume throughout this work that Alice (A) wants to encrypt a message $m \in \mathbb{Z}_{>0}$ intended for Bob (B). We denote Bob's unique identity, for example his email-address, by ID_B and his public key by \mathfrak{b}

and p_2 , and the workload for the KGC is consequently very high. Furthermore, since one chooses $p_i - 1$ smooth and uses Pohlig-Hellman's simplification together with Shank's Baby-Step Giant-Step algorithm, the time needed for generating k user keys is proportional to k .

In contrast, we use two different subexponential algorithms for the key generation. After the initial computation of relations over the factor bases, the workload for each individual key generation is very modest. For the computation of discrete logarithms in the class group of the maximal order, $Cl(\Delta_1)$, we use an analogue of the Self-Initializing Quadratic Sieve (SIQS) factoring algorithm [14,13] and for the computation of discrete logarithms in \mathbb{F}_p^* we use the Special Number Field Sieve, which recently was used for the solution of McCurley's challenge [24].

This paper is organized as follows: in Section 2 we provide the necessary background and notation for non-maximal imaginary quadratic orders. The next section contains the discrete logarithm algorithm for arbitrary non-maximal imaginary quadratic orders, and in Section 4 we present our new non-interactive public key cryptosystem. In order to save space, the proofs of most results have been omitted. These proofs, as well as computational results, will be given in the full paper [10].

2 Non-maximal Imaginary Quadratic Orders

The basic notions of imaginary quadratic number fields can be found in [1,2]. For a more comprehensive treatment of the relationship between maximal and non-maximal orders we refer to [5,9,12].

Let \mathcal{O}_{Δ_f} denote the non-maximal quadratic order of discriminant $\Delta_f = \Delta_1 f^2$ with conductor f , and let \mathcal{O}_{Δ_1} denote the corresponding maximal order. When the conductor is prime, we will use \mathcal{O}_{Δ_p} and Δ_p . By $Cl(\Delta_f)$ and $Cl(\Delta_1)$ we denote the ideal class groups of \mathcal{O}_{Δ_f} and \mathcal{O}_{Δ_1} , respectively. The class numbers $h(\Delta_f)$ and $h(\Delta_1)$ are the orders of these groups. Lower-case Gothic letters $\mathfrak{a}, \mathfrak{b}, \dots$ denote ideals in \mathcal{O}_{Δ_f} and upper-case Gothic letters denote ideals in \mathcal{O}_{Δ_1} . Ideal equivalence is denoted by $\mathfrak{a} \sim \mathfrak{b}$, and the class of all ideals equivalent to \mathfrak{a} is denoted by $[\mathfrak{a}]$. Throughout, we will use Δ without subscript to denote the discriminant of an arbitrary quadratic order, maximal or non-maximal.

Our cryptosystem makes use of the relationship between a non-maximal order of conductor f and its corresponding maximal order. Any non-maximal order can be represented as $\mathcal{O}_{\Delta_f} = \mathbb{Z} + f\mathcal{O}_{\Delta_1}$. If $h(\Delta_1) = 1$, then \mathcal{O}_{Δ_f} is called a totally non-maximal order. An integral ideal \mathfrak{a} is called prime to f if $\gcd(\mathcal{N}(\mathfrak{a}), f) = 1$. It is well-known that all \mathcal{O}_{Δ_f} -ideals prime to the conductor are invertible, and in every ideal equivalence class there is an ideal which is prime to any given number. We denote the principal \mathcal{O}_{Δ_f} -ideals prime to f by $\mathcal{P}_{\Delta_f}(f)$ and all fractional ideals which are prime to f by $\mathcal{I}_{\Delta_f}(f)$. There is an isomorphism

$$\mathcal{I}_{\Delta_f}(f) / \mathcal{P}_{\Delta_f}(f) \simeq \mathcal{I}_{\Delta_f} / \mathcal{P}_{\Delta_f} = Cl(\Delta_f) , \quad (1)$$

so we can “ignore” the ideals which are not prime to the conductor if we are only interested in the class group $Cl(\Delta_f)$.

There is an isomorphism between the group of \mathcal{O}_{Δ_f} -ideals which are prime to f and the group of \mathcal{O}_{Δ_1} -ideals which are prime to f , denoted by $\mathcal{I}_{\Delta_f}(f)$, and $\mathcal{I}_{\Delta_1}(f)$, respectively.

Proposition 1. *Let \mathcal{O}_{Δ_f} be an order of conductor f in an imaginary quadratic field $\mathbb{Q}(\sqrt{\Delta_1})$ with maximal order \mathcal{O}_{Δ_1} .*

- (i.) *If $\mathfrak{A} \in \mathcal{I}_{\Delta_1}(f)$, then $\mathfrak{a} = \mathfrak{A} \cap \mathcal{O}_{\Delta_f} \in \mathcal{I}_{\Delta_f}(f)$ and $\mathcal{N}(\mathfrak{A}) = \mathcal{N}(\mathfrak{a})$.*
- (ii.) *If $\mathfrak{a} \in \mathcal{I}_{\Delta_f}(f)$, then $\mathfrak{A} = \mathfrak{a}\mathcal{O}_{\Delta_1} \in \mathcal{I}_{\Delta_1}(f)$ and $\mathcal{N}(\mathfrak{a}) = \mathcal{N}(\mathfrak{A})$.*
- (iii.) *The map $\varphi : \mathfrak{A} \mapsto \mathfrak{A} \cap \mathcal{O}_{\Delta_f}$ induces an isomorphism $\mathcal{I}_{\Delta_1}(f) \xrightarrow{\sim} \mathcal{I}_{\Delta_f}(f)$.
The inverse of this map is $\varphi^{-1} : \mathfrak{a} \mapsto \mathfrak{a}\mathcal{O}_{\Delta_1}$.*

Thus we are able to switch to and from ideals in the maximal and non-maximal orders via the map φ . The algorithms `GoToMaxOrder`(\mathfrak{a}, f) to compute φ^{-1} and `GoToNonMaxOrder`(\mathfrak{A}, f) to compute φ can be found in [9]. If $\mathfrak{a} = a\mathbb{Z} + \frac{b+\sqrt{\Delta_f}}{2}\mathbb{Z} = (a, b)$ and $\mathfrak{A} = A\mathbb{Z} + \frac{B+\sqrt{\Delta_1}}{2}\mathbb{Z} = (A, B)$ are reduced ideals, then these algorithms need $O(\log(|\Delta_1|)^2)$ and $O(\log(|\Delta_f|)^2)$ bit-operations respectively.

It is important to note that the isomorphism φ is between the *ideal groups* $\mathcal{I}_{\Delta_1}(f)$ and $\mathcal{I}_{\Delta_f}(f)$ and *not the class groups*. If, for $\mathfrak{A}, \mathfrak{B} \in \mathcal{I}_{\Delta_1}(f)$ we have $\mathfrak{A} \sim \mathfrak{B}$, it is not necessarily true that $\varphi(\mathfrak{A}) \sim \varphi(\mathfrak{B})$. On the other hand, equivalence *does* hold under φ^{-1} . More precisely we have the following:

Proposition 2. *The isomorphism φ^{-1} induces a surjective homomorphism $\phi_{Cl}^{-1} : Cl(\Delta_f) \rightarrow Cl(\Delta_1)$, where $[\mathfrak{a}] \mapsto [\varphi^{-1}(\mathfrak{a})]$.*

We now focus on the kernel $\text{Ker}(\phi_{Cl}^{-1})$ of this map, which will turn out to be of central importance for the computation of discrete logarithms in $Cl(\Delta_f)$. In particular, we will need to compute discrete logarithms of elements in $\text{Ker}(\phi_{Cl}^{-1})$. Representing elements of $\text{Ker}(\phi_{Cl}^{-1})$ as ideal equivalence classes is completely inadequate for this purpose since we would have to compute discrete logarithms in $Cl(\Delta_f)$. Fortunately, there exists an alternative representation which allows us to reduce the problem of computing discrete logarithms in $\text{Ker}(\phi_{Cl}^{-1})$ to that in a small number of finite fields.

Proposition 3. *The map $\psi : (\mathcal{O}_{\Delta_1}/f\mathcal{O}_{\Delta_1})^* \rightarrow \text{Ker}(\phi_{Cl}^{-1})$, $[\alpha] \mapsto [\varphi(\alpha\mathcal{O}_{\Delta_1})]$, is a surjective homomorphism.*

This homomorphism suggests the following representation for ideal classes in the kernel:

Definition 1. *Let $[\alpha] = [x + y\omega] \in (\mathcal{O}_{\Delta_1}/f\mathcal{O}_{\Delta_1})^*$ and let $\mathfrak{a} \sim \varphi(\alpha\mathcal{O}_{\Delta_1})$ be a reduced \mathcal{O}_{Δ_f} -ideal whose equivalence class lies in $\text{Ker}(\phi_{Cl}^{-1})$. Then the pair (x, y) is called a generator representation for the equivalence class $[\mathfrak{a}]$.*

Remark 1. Note that this generator representation (x, y) for the class of \mathfrak{a} is *not unique*. It is easy to see that (kx, ky) , $k \in (\mathbb{Z}/f\mathbb{Z})^*$, is also a generator representation for the class of \mathfrak{a} . This means that we have $\mathfrak{a} \sim \varphi((x + y\omega)\mathcal{O}_{\Delta_1}) \sim \varphi((kx + ky\omega)\mathcal{O}_{\Delta_1})$. In other words, $\text{Ker}(\phi_{Cl}^{-1}) \cong (\mathcal{O}_{\Delta_1}/f\mathcal{O}_{\Delta_1})^*/i((\mathbb{Z}/f\mathbb{Z})^*)$, where i denotes the natural embedding of $\mathbb{Z}/f\mathbb{Z}$ into $(\mathcal{O}_{\Delta_1}/f\mathcal{O}_{\Delta_1})^*$, as illustrated by the exact sequence (7.27) in [5, p.147].

Our reduction of the discrete logarithm problem in $Cl(\Delta_f)$ to $Cl(\Delta_1)$ and finite fields requires computing various preimages of elements in $\text{Ker}(\phi_{Cl}^{-1})$ under the map ψ . Algorithm 1 (Std2Gen) accomplishes this task. The algorithm Reduce reduces an ideal \mathfrak{A} given in standard representation and simultaneously computes a reducing number $\gamma \in \mathcal{O}_{\Delta_1}$ of the form $(\bar{x} + \bar{y}\sqrt{\Delta_1})/2$ such that \mathfrak{A}/γ is reduced (see, for example, [14, Algorithm 2.6, p.16]).

Algorithm 1 Std2Gen

Input: The standard representation (a, b) of a reduced \mathcal{O}_{Δ_f} -ideal $\mathfrak{a} = a\mathbb{Z} + \frac{b+\sqrt{\Delta_f}}{2}\mathbb{Z}$ representing a class in $\text{Ker}(\phi_{Cl}^{-1})$, and the conductor f .

Output: A generator representation (x, y) of the class $[\mathfrak{a}] \in \text{Ker}(\phi_{Cl}^{-1})$.

```

(A, B) ← GoToMaxOrder( $\mathfrak{a}$ ,  $f$ )
( $\mathfrak{G}$ ,  $\gamma$ ) ← Reduce( $A, B$ )
if  $\mathfrak{G} \not\sim \mathcal{O}_{\Delta_1}$  then
    return('Error!  $\mathfrak{a} \notin \text{Ker}(\phi_{Cl}^{-1})$ !')
end if
if  $\Delta_1 \equiv 0 \pmod{4}$  then
     $x \leftarrow \bar{x}/2 \pmod{f}$ 
     $y \leftarrow \bar{y}/2 \pmod{f}$ 
else
     $x \leftarrow (\bar{x} - \bar{y})/2 \pmod{f}$ 
     $y \leftarrow \bar{y} \pmod{f}$ 
end if
return( $(x, y)$ )

```

3 The DLP for Arbitrary $Cl(\Delta_f)$

In this section we generalize the result from [12]. We show that given the conductor f and its prime factorization one can reduce the DLP in an arbitrary $Cl(\Delta_f)$ to the DLP in various smaller groups. More precisely, we first show that the computation of discrete logarithms in $Cl(\Delta_f)$ can be reduced to the computation of discrete logarithms in the class group $Cl(\Delta_1)$ of the maximal order and the computation of discrete logarithms in $\text{Ker}(\phi_{Cl}^{-1})$. Furthermore, we show that the latter problem boils down to the computation of discrete logarithms in a small number of finite fields.

It should be noted that our method here is in essence a special case of the more general methods employed by Cohen et al. to compute discrete logarithms in ray class groups [3]. The class group of a non-maximal order in any number field, not only degree 2, can be viewed as a ray class group of the maximal order, where the modulus is simply an integer, the conductor of the non-maximal order. Our exposition here is a reformulation of these results in terms of the simpler, special case of non-maximal orders using the language of [12]. In addition, we prove that the reduction of the DLP in $Cl(\Delta_f)$ to computing discrete logarithm computations in $Cl(\Delta_1)$ and a small number of finite fields is of polynomial complexity.

We start with an algorithm which reduces the DLP in $Cl(\Delta_f)$ to the DLP in $Cl(\Delta_1)$ and $\text{Ker}(\phi_{Cl}^{-1})$. Since the map $\psi : (\mathcal{O}_{\Delta_1}/f\mathcal{O}_{\Delta_1})^* \rightarrow \text{Ker}(\phi_{Cl}^{-1})$ given in Proposition 3 induces the isomorphism $\text{Ker}(\phi_{Cl}^{-1}) \cong (\mathcal{O}_{\Delta_1}/f\mathcal{O}_{\Delta_1})^*/i((\mathbb{Z}/f\mathbb{Z})^*)$, we will reduce the latter DLP to computations in $(\mathcal{O}_{\Delta_1}/f\mathcal{O}_{\Delta_1})^*$. Thus, our algorithm makes use of the following two methods:

- **DLPinCl**($\mathfrak{G}, \mathfrak{A}$)
Accepts two reduced \mathcal{O}_{Δ_1} -ideals $\mathfrak{G}, \mathfrak{A}$ as input and returns $x \in \mathbb{Z}$ with $0 \leq x < h(\Delta_1)$ such that $\mathfrak{G}^x \sim \mathfrak{A}$, or $x = -1$ if no such x exists.
- **DLPinKerphi**($\gamma, \alpha, |\text{Ker}(\phi_{Cl}^{-1})|$)
Accepts two generator representations γ, α of classes in $\text{Ker}(\phi_{Cl}^{-1})$ such that $[\gamma], [\alpha] \in (\mathcal{O}_{\Delta_1}/f\mathcal{O}_{\Delta_1})^*$ as input and returns $x \in \mathbb{Z}$ with $0 \leq x < |\text{Ker}(\phi_{Cl}^{-1})|$ such that $\psi([\gamma])^x = \psi([\alpha])$ in $\text{Ker}(\phi_{Cl}^{-1})$, or $x = -1$ if no such x exists.

Furthermore, we assume that $h(\Delta_1)$ is known. This is no practical restriction, since the best currently known algorithm [14,13] for computing discrete logarithms in $Cl(\Delta_1)$ needs to compute $h(\Delta_1)$ and the group structure of $Cl(\Delta_1)$ before the actual DL-computation starts. Secondly, if there were any other algorithm **DLPinCl** with the above properties, then one could use it to compute $h(\Delta_1)$, as shown in the full paper [10].

Algorithm 2 (**ReduceDLP**) reduces the DLP in $Cl(\Delta_f)$ to the DLP in $Cl(\Delta_1)$ and $\text{Ker}(\phi_{Cl}^{-1}) \cong (\mathcal{O}_{\Delta_1}/f\mathcal{O}_{\Delta_1})^*/i((\mathbb{Z}/f\mathbb{Z})^*)$. The proof of correctness can be found in the full version of the paper [10].

Proposition 4. *Given the conductor f , the class number $h(\Delta_1)$ and the order of the kernel $|\text{Ker}(\phi_{Cl}^{-1})|$ one can reduce the DLP in $Cl(\Delta_f)$ in $O(\log(|\Delta_f|)^3)$ bit-operations to the DLP in $Cl(\Delta_1)$ and $\text{Ker}(\phi_{Cl}^{-1})$.*

Thus, in order to compute discrete logarithms in $Cl(\Delta_f)$, we need efficient algorithms for computing discrete logarithms in $Cl(\Delta_1)$ and $\text{Ker}(\phi_{Cl}^{-1})$. The subexponential algorithm described in [13, Algorithm 3.3] is the most efficient algorithm known for computing discrete logarithms in $Cl(\Delta_1)$. We now consider the DLP in $\text{Ker}(\phi_{Cl}^{-1}) \cong (\mathcal{O}_{\Delta_1}/f\mathcal{O}_{\Delta_1})^*/i((\mathbb{Z}/f\mathbb{Z})^*)$ more closely.

By the Chinese Remainder Theorem (see, for example, [15, p.11]), the DLP in $(\mathcal{O}_{\Delta_1}/f\mathcal{O}_{\Delta_1})^*/i((\mathbb{Z}/f\mathbb{Z})^*)$ boils down to DLPs in $(\mathcal{O}_{\Delta_1}/p_i^{e_i}\mathcal{O}_{\Delta_1})^*/i((\mathbb{Z}/p_i^{e_i}\mathbb{Z})^*)$ for prime powers $p_i^{e_i}$, where $f = \prod p_i^{e_i}$. Furthermore, this problem can be efficiently reduced to the prime case $(\mathcal{O}_{\Delta_1}/p_i\mathcal{O}_{\Delta_1})^*/i(\mathbb{F}_{p_i}^*)$. We give an algorithm (**ReducePe2P**) for this reduction in the full version of the paper [10].

Algorithm 2 ReduceDLP

Input: Two reduced \mathcal{O}_{Δ_f} -ideals $\mathfrak{g}, \mathfrak{a}$, the conductor f , the class number $h(\Delta_1)$, and the order of the kernel $|\text{Ker}(\phi_{Cl}^{-1})| = \frac{f}{[\mathcal{O}_{\Delta_1}^* : \mathcal{O}_{\Delta_f}^*]} \prod_{p|f} (1 - \frac{(\Delta/p)}{p})$

Output: The discrete logarithm x , such that $\mathfrak{g}^x \sim \mathfrak{a}$, with $0 \leq x < h(\Delta_f)$, or $x = -1$, if no such x exists.

```

{Compute DL in  $Cl(\Delta_1)$ }
 $\mathfrak{G} \leftarrow \text{GoToMaxOrder}(\mathfrak{g}, f)$ 
 $\mathfrak{A} \leftarrow \text{GoToMaxOrder}(\mathfrak{a}, f)$ 
 $x_1 \leftarrow \text{DLPinCl}(\mathfrak{G}, \mathfrak{A})$ 
if  $x_1 = -1$  then
  return(-1)
end if
{Compute DL in  $(\mathcal{O}_{\Delta_1}/f\mathcal{O}_{\Delta_1})^*$ }
 $\alpha \leftarrow \text{Std2Gen}(\mathfrak{a}/\mathfrak{g}^{x_1}, f)$ 
 $\gamma \leftarrow \text{Std2Gen}(\mathfrak{g}^{h(\Delta_1)}, f)$ 
 $c \leftarrow \text{DLPinKerphi}(\gamma, \alpha, |\text{Ker}(\phi_{Cl}^{-1})|)$ 
if  $c = -1$  then
  return(-1)
end if
{Combine partial results to get DL in  $Cl(\Delta_f^2)$ }
 $x \leftarrow c \cdot h(\Delta_1) + x_1$ 
return( $x$ )

```

Proposition 5. *The DLP in $(\mathcal{O}_{\Delta_1}/p^e\mathcal{O}_{\Delta_1})^*/i((\mathbb{Z}/p^e\mathbb{Z})^*)$ can be reduced in $O(e \cdot (\log p)^3)$ bit-operations to $2e$ DL-computations in $(\mathcal{O}_{\Delta_1}/p\mathcal{O}_{\Delta_1})^*/i(\mathbb{F}_p^*)$.*

Corollary 1. *If $e = O((\log p)^\alpha)$ for some $\alpha = O(1)$, then the DLP in $(\mathcal{O}_{\Delta_1}/p^e\mathcal{O}_{\Delta_1})^*/i((\mathbb{Z}/p^e\mathbb{Z})^*)$ can be reduced in polynomial time (in $\log p$) to the DLP in $(\mathcal{O}_{\Delta_1}/p\mathcal{O}_{\Delta_1})^*/i(\mathbb{F}_p^*)$.*

Using ReduceDLP and ReducePe2P allows us to reduce the DLP in $Cl(\Delta_f)$ to DLPs in $Cl(\Delta_1)$ and $(\mathcal{O}_{\Delta_1}/p\mathcal{O}_{\Delta_1})^*/i(\mathbb{F}_p^*)$. As shown in [12,11], $(\mathcal{O}_{\Delta_1}/p\mathcal{O}_{\Delta_1})^*$ is isomorphic to either $\mathbb{F}_p^* \times \mathbb{F}_p^*$ or $\mathbb{F}_{p^2}^*$, depending how p splits in \mathcal{O}_{Δ_1} . This immediately leads to the central result of this section.

Theorem 1. *If the prime factorization of the conductor $f = \prod_{i=1}^k p_i^{e_i}$ is known and $e_i = O((\log p_i)^\alpha)$ for some $\alpha = O(1)$ then one can reduce the discrete logarithm problem in $Cl(\Delta_f)$ in polynomial time (in $\log \Delta_f$) to the computation of logarithms in $Cl(\Delta_1)$ and the following groups ($1 \leq i \leq k$):*

$$\begin{aligned} & \mathbb{F}_{p_i}^*, \text{ if } \left(\frac{\Delta_1}{p_i} \right) \in \{0, 1\} \\ & \mathbb{F}_{p_i^2}^*, \text{ if } \left(\frac{\Delta_1}{p_i} \right) = -1. \end{aligned}$$

Proof. If the conductor f and its prime factorization are known, then one can use ReduceDLP (Algorithm 2) to reduce the DLP in $Cl(\Delta_f)$ to the DLP in $Cl(\Delta_1)$ and $\text{Ker}(\phi_{Cl}^{-1})$. By Proposition 4 this is possible in polynomial time in $\log \Delta_f$. By the Chinese Remainder Theorem (using the known factorization of f) the

DLP in $\text{Ker}(\phi_{Cl}^{-1}) \cong (\mathcal{O}_{\Delta_1}/f\mathcal{O}_{\Delta_1})^*/i((\mathbb{Z}/f\mathbb{Z})^*)$ is nothing more than the DLP in groups of the form $(\mathcal{O}_{\Delta_1}/p_i^{e_i}\mathcal{O}_{\Delta_1})^*/i((\mathbb{Z}/p_i^{e_i}\mathbb{Z})^*)$, which can, using ReducePe2P (from [10]) and Corollary 1, be reduced in polynomial time (in $\log p_i$) to the DLP in $(\mathcal{O}_{\Delta_1}/p_i\mathcal{O}_{\Delta_1})^*/i(\mathbb{F}_{p_i}^*)$, because e_i is assumed to be polynomial in $\log p_i$.

It remains to show how one reduces the discrete logarithm problem in $(\mathcal{O}_{\Delta_1}/p\mathcal{O}_{\Delta_1})^*/i(\mathbb{F}_p^*)$ to discrete logarithm problems in \mathbb{F}_p^* or $\mathbb{F}_{p^2}^*$. Suppose we have two representatives γ, α of classes in $(\mathcal{O}_{\Delta_1}/p\mathcal{O}_{\Delta_1})^*$ for which we want to compute the discrete logarithm c such that $[\gamma]^c \equiv [\alpha]$ in $(\mathcal{O}_{\Delta_1}/p\mathcal{O}_{\Delta_1})^*/i(\mathbb{F}_p^*)$. In the inert case $(\Delta_1/p) = -1$, where $(\mathcal{O}_{\Delta_1}/p\mathcal{O}_{\Delta_1})^* \cong \mathbb{F}_{p^2}^*$, we have $(\mathcal{O}_{\Delta_1}/p\mathcal{O}_{\Delta_1})^*/i(\mathbb{F}_p^*) \cong \mathbb{F}_{p^2}^*/i(\mathbb{F}_p^*)$. It is well-known that there always exists a surjective homomorphism from $\mathbb{F}_{p^2}^*$ to $\mathbb{F}_{p^2}^*/i(\mathbb{F}_p^*)$. Thus, we first solve the DLP $\gamma^{c'} \equiv \alpha \pmod{p\mathcal{O}_{\Delta_1}}$ by simply solving the corresponding DLP in $\mathbb{F}_{p^2}^*$. Taking $c \equiv c' \pmod{p+1}$ yields the required solution to the DLP $[\gamma]^c \equiv [\alpha]$ in $(\mathcal{O}_{\Delta_1}/p\mathcal{O}_{\Delta_1})^*/i(\mathbb{F}_p^*)$.

We now restrict our attention to the split case $(\Delta_1/p) = 1$, where we have $(\mathcal{O}_{\Delta_1}/p\mathcal{O}_{\Delta_1})^* \cong \mathbb{F}_p^* \times \mathbb{F}_p^*$. The element $\gamma = (x_1, y_1)$ maps to $(x_1 \bmod p, y_1 \bmod p) \in \mathbb{F}_p^* \times \mathbb{F}_p^*$ and similarly $\alpha = (x_2, y_2)$ maps to $(x_2 \bmod p, y_2 \bmod p)$. The DLP in $(\mathcal{O}_{\Delta_1}/p\mathcal{O}_{\Delta_1})^*/i(\mathbb{F}_p^*)$ becomes

$$(x_1, y_1)^c \equiv l(x_2, y_2) \quad (\text{in } \mathbb{F}_p^* \times \mathbb{F}_p^*)$$

which in turn yields the simultaneous DLP's

$$x_1^c \equiv lx_2 \pmod{p}, \quad y_1^c \equiv ly_2 \pmod{p}.$$

Since these two DLP's must be solved for the same c and l , we can combine them and obtain the single DLP in \mathbb{F}_p^*

$$\left(\frac{x_1}{y_1}\right)^c \equiv \left(\frac{x_2}{y_2}\right) \pmod{p}$$

from which we can find the desired value of c .

As noted in [8], this simple strategy can be used to improve the general maps from [12, 11]; it is shown that in this case there not only exists a surjective homomorphism $\mathbb{F}_p^* \times \mathbb{F}_p^* \rightarrow \text{Ker}(\phi_{Cl}^{-1})$, but even an efficiently computable isomorphism $\mathbb{F}_p^* \cong \text{Ker}(\phi_{Cl}^{-1})$. \square

Note that the central result of [12] now is nothing more than an immediate corollary.

3.1 Example

We illustrate the reduction of discrete logarithm computations in $Cl(\Delta_f)$ via a small example. Suppose $\Delta_1 = -1019$, $f = 23$, and $\Delta_f = \Delta_1 f^2 = -539051$. In this case, both $Cl(\Delta_f)$ and $Cl(\Delta_1)$ are cyclic with $h(\Delta_1) = 13$ and $h(\Delta_f) = h(\Delta_1)(23 - 1) = 286$. The equivalence class represented by the reduced ideal

$$\mathfrak{g} = 15\mathbb{Z} + \frac{-7 + \sqrt{-539051}}{2}\mathbb{Z} = (15, -7)$$

generates $Cl(\Delta_f)$.

Suppose we wish to compute the discrete logarithm of $[\mathfrak{a}]$ with respect to the base $[\mathfrak{g}]$ in $Cl(\Delta_f)$, where

$$\mathfrak{a} = 11\mathbb{Z} + \frac{9 + \sqrt{-539051}}{2}\mathbb{Z} = (11, 9) .$$

That is, we want to find x such that $\mathfrak{g}^x \sim \mathfrak{a}$. Since \mathfrak{g} generates $Cl(\Delta_f)$, we know that such an x exists. Following ReduceDLP (Algorithm 2), we first compute $[\mathfrak{G}] = [\phi_{Cl}^{-1}(\mathfrak{g})]$ and $[\mathfrak{A}] = [\phi_{Cl}^{-1}(\mathfrak{a})]$, and solve the discrete logarithm problem

$$\mathfrak{G}^{x_1} \sim \mathfrak{A}$$

in $Cl(\Delta_1)$. We have $\mathfrak{G} = 15\mathbb{Z} + \frac{1+\sqrt{-1019}}{2}\mathbb{Z} = (15, 1)$, $\mathfrak{A} = (11, 9)$, and we easily compute $x_1 = 9$.

At this point we know that x has the form $x = c \cdot h(\Delta_1) + x_1 = 13c + 9$, and it remains to compute c . Again following ReduceDLP (Algorithm 2), we compute generator representations α, γ of $[\alpha], [\gamma] \in (\mathcal{O}_{\Delta_1}/f\mathcal{O}_{\Delta_1})^*$ such that $\psi([\alpha]) = [\mathfrak{a}/\mathfrak{g}^{x_1}]$ and $\psi([\gamma]) = [\mathfrak{g}^{h(\Delta_1)}]$. Following Std2Gen (Algorithm 1), we first compute

$$\mathfrak{b} \sim \mathfrak{a}/\mathfrak{g}^{x_1} \sim \mathfrak{a}/\mathfrak{g}^9 = (311, 277)$$

and

$$\mathfrak{c} \sim \mathfrak{g}^{h(\Delta_1)} \sim \mathfrak{g}^{13} = (297, 295) .$$

To find α and γ we compute the principal ideals $\mathfrak{B} = \varphi^{-1}(\mathfrak{b})$ and $\mathfrak{C} = \varphi^{-1}(\mathfrak{c})$, and reduce them while simultaneously computing their modulo $f\mathcal{O}_{\Delta_1}$ reduced generators, which we take as α and γ . We obtain $\mathfrak{B} = (311, -15) = (\alpha)$ and $\mathfrak{C} = (297, -13) = (\gamma)$ where

$$\alpha = -8 + 1\omega, \quad \gamma = -7 + 1\omega$$

and $\omega = \frac{1+\sqrt{-1019}}{2}$.

To compute c , we need to solve the discrete logarithm problem

$$[\gamma]^c \equiv [\alpha] \quad (\text{in } \text{Ker}(\phi_{Cl}^{-1}) \cong (\mathcal{O}_{\Delta_1}/f\mathcal{O}_{\Delta_1})^*/i((\mathbb{Z}/f\mathbb{Z})^*)) .$$

For this example, we have $(\Delta_1/f) = (-1019/23) = 1$, and thus $(\mathcal{O}_{\Delta_1}/f\mathcal{O}_{\Delta_1})^* \simeq \mathbb{F}_{23}^* \times \mathbb{F}_{23}^*$ by [12, Lemma 8]. Since $\omega \equiv 14 \pmod{23}$ and $\bar{\omega} \equiv 10 \pmod{23}$, we obtain

$$\gamma \mapsto (-7 + 1\omega \bmod 23, -7 + 1\bar{\omega} \bmod 23) = (7, 3) \in \mathbb{F}_{23}^* \times \mathbb{F}_{23}^*$$

and

$$\alpha \mapsto (-8 + 1\omega \bmod 23, -8 + 1\bar{\omega} \bmod 23) = (6, 2) \in \mathbb{F}_{23}^* \times \mathbb{F}_{23}^* .$$

Since $\text{Ker}(\phi_{Cl}^{-1}) \cong (\mathbb{F}_p^* \times \mathbb{F}_p^*)/i(\mathbb{F}_p^*)$, we need to find c by solving the discrete logarithm problem $(7, 3)^c = l(6, 2)$ in $\mathbb{F}_{23}^* \times \mathbb{F}_{23}^*$ for every $l \in \mathbb{F}_{23}^*$. This yields

$$7^c \equiv 6l \pmod{23}, \quad 3^c \equiv 2l \pmod{23},$$

and we combine these two discrete logarithm problems to obtain one discrete logarithm problem in \mathbb{F}_{23}^* :

$$(7/3)^c \equiv (6/2) \pmod{23} \rightarrow 10^c \equiv 3 \pmod{23} .$$

Solving yields $c = 20$, and finally $x = 13 \cdot 20 + 9 = 269$. It is easy to verify that x is indeed the desired discrete logarithm: simply compute the reduced ideal \mathfrak{g}^{269} and verify that it is equal to the reduced ideal \mathfrak{a} .

4 Towards Practical Non-interactive Cryptosystems

Before we explain our system setup we list the crucial properties:

Required Properties

1. The discrete logarithm problem (DLP) in $Cl(\Delta_p)$ *without* knowing the factorization of $\Delta_p = \Delta_1 p^2$ is infeasible. To determine bounds for Δ_1 and p , we make use of the heuristic model from [7], which is a refinement of Lenstra and Verheul's approach [16], since it also takes into account the asymptotically vanishing $o(1)$ -part in subexponential algorithms. We will now derive bounds for the parameters such that an attacker would need to spend about 90,000 MIPS years to break the system. This approximately amounts to a ten-fold higher workload than the recent factorization of RSA155 and hence corresponds to the very minimum requirements. The estimates in [7, Table 3] state that Δ_p should have at least 576,667,423 bits to prevent factoring Δ_p with the GNFS, factoring Δ_p with ECM and computing discrete logarithms in $Cl(\Delta_p)$ with the SIQS-analogue [14], respectively.
 - 1.1 Δ_p is large enough that using the subexponential algorithm from [13] to directly compute discrete logarithms in $Cl(\Delta_p)$ is infeasible. $\Delta_p > 2^{423}$ implies an expected workload of more than 90,000 MIPS years.
 - 1.2 Δ_p cannot be factored to reduce the DLP to DLPs in $Cl(\Delta_1)$ and \mathbb{F}_p^* (or $\mathbb{F}_{p^2}^*$).
 - 1.2.1 Δ_p is large enough so that the Number Field Sieve would need more than 90,000 MIPS years. This yields $\Delta_p > 2^{576}$.
 - 1.2.2 Δ_1 and p are large enough that it would take more than 90,000 MIPS years to find them with the Elliptic Curve Method. This implies $\Delta_1, p > 2^{222}$.
2. Δ_1, p must be small enough to enable the KGC to compute discrete logarithms in $Cl(\Delta_1)$ and \mathbb{F}_p^* using subexponential algorithms. $\Delta_1, p < 2^{300}$ seems to be feasible.
3. $Cl(\Delta_p)$ must be cyclic.

It is easy to see that the following setup satisfies *all* above requirements.

System Setup

1. The KGC randomly chooses a prime $q \equiv 3 \pmod{4}$, $q > 2^{260}$, sets $\Delta_1 = -q$ and computes $h(\Delta_1)$ and the group structure of $Cl(\Delta_1)$ with the algorithm from [14]. The Cohen-Lenstra heuristics [4] suggest that $Cl(\Delta_1)$ is cyclic with probability > 0.97 . If $Cl(\Delta_1)$ is not cyclic, the KGC selects another prime q until it is cyclic.
2. The KGC chooses a prime $p > 2^{260}$ with $(\Delta_1/p) = 1$ and $\gcd(p-1, h(\Delta_1)) = 1$ such that the SNFS can be applied as in [24], and computes $\Delta_p = \Delta_1 p^2$. The gcd condition ensures that $Cl(\Delta_p)$ is cyclic.
3. The KGC computes a generator \mathfrak{g} of $Cl(\Delta_p)$ and publishes it together with Δ_p .

Given a generator \mathfrak{G} of $Cl(\Delta_1)$, which the KGC can easily obtain during the computation of $Cl(\Delta_1)$ [14, Algorithm 6.1], it is also easy in practice to find a generator \mathfrak{g} of $Cl(\Delta_p)$ with the additional property that $\phi_{Cl}^{-1}(\mathfrak{g}) = \mathfrak{G}$. The KGC repeatedly selects random values of $\alpha \in \mathcal{O}_{\Delta_1}$ and takes the first $\mathfrak{g} = \phi(\alpha\mathfrak{G})$ such that $\mathfrak{g}^{h(\Delta_p)/d_i} \not\sim \mathcal{O}_{\Delta_p}$ for any positive divisor d_i of $h(\Delta_p)$. Although $h(\Delta_p)$ is approximately as large as $\sqrt{|\Delta_p|}$, in practice it has sufficiently many small factors that this condition can be verified with high probability.

User Registration

1. Bob requests the public key \mathfrak{b} corresponding to his identity ID_B at the KGC.
2. The KGC verifies Bob's identity, for example, using a passport, and starts with the key generation.
3. The KGC computes the 128-bit hash $id = h(ID_B)$ using, for example, MD5 [22], of Bob's identity and embeds id into a group element of $Cl(\Delta_p)$ by taking the largest prime $p_B \leq id$, for which $(\Delta_p/p_B) = 1$ and computing the prime ideal $\mathfrak{b} = p_B\mathbb{Z} + \frac{b_B + \sqrt{\Delta_p}}{2}$, where b_B is the uniquely determined square root of $\Delta_p \bmod 4p_B$ with $0 \leq b_B \leq p_B$. Note that \mathfrak{b} is already reduced, since $\sqrt{|\Delta_p|} > 2^{128} > p_B$. If the KGC recognizes that \mathfrak{b} is already assigned to another user it will ask Bob to choose another identity, for example, his postal address.
4. Finally, the KGC computes the discrete logarithm b such that $\mathfrak{g}^b \sim \mathfrak{b}$ using the secret knowledge of the conductor p and the reduction procedure described in the Section 3, and returns b to Bob.

As soon as all users are registered this way the KGC can destroy the factorization of Δ_p and cease to exist. The users can obtain any other user's *authentic* public key simply by hashing that user's identity and computing the largest prime ideal whose norm is less than the hash value. Each user has a public/private key-pair (\mathfrak{a}, a) with $\mathfrak{a} \sim \mathfrak{g}^a$, so discrete logarithm-based protocols such as Diffie-Hellman or ElGamal can be directly applied in the class group $Cl(\Delta_p)$.

Preliminary experiments, together with computational experience using the subexponential algorithms from [13] and [24], indicate that a KGC with modest computational resources, for example, a small network of Pentium processors, should be able to set up a key distribution system using $p, q \approx 2^{300}$ at least. For such an example, we estimate that after a precomputation of about 3 days on a cluster of 16 550 Mhz Pentiums III's for computing the class group $Cl(\Delta_1)$, each user registration would take about 1 day on a single 550 Mhz Pentiums III, the vast majority of this time being spent on the computation of discrete logarithms in \mathbb{F}_p^* . However, adding more machines to the cluster yields a linear speedup in both the precomputation stage and part of the user registration stage. Thus, although this level of complexity is far from ideal, unlike the case of $(\mathbb{Z}/n\mathbb{Z})^*$ it is at least possible to set up noninteractive systems with secure parameters in $Cl(\Delta_p)$. More detailed computational results will appear in the full version of the paper [10].

References

1. Z.I. Borevich and I.R. Shafarevich. *Number Theory*. Academic Press, New York, 1966.
2. H. Cohen. *A Course in Computational Algebraic Number Theory*. Springer-Verlag, Berlin, 1993.
3. H. Cohen, F. Diaz y Diaz, and M. Olivier. Computing ray class groups, conductors, and discriminants. *Math. Comp.*, 67(222):773–795, 1998.
4. H. Cohen and H.W. Lenstra, Jr. Heuristics on class groups of number fields. In *Number Theory, Lecture notes in Math.*, volume 1068, pages 33–62. Springer-Verlag, New York, 1983.
5. D.A. Cox. *Primes of the form $x^2 + ny^2$* . John Wiley & Sons, New York, 1989.
6. D. Hühnlein. Efficient implementation of cryptosystems based on non-maximal imaginary quadratic orders. In *Selected Areas in Cryptography - SAC'99*, volume 1758 of *LNCS*, pages 150–167, 1999.
7. D. Hühnlein. Quadratic orders for NESSIE – Overview and parameter sizes of three public key families. Technical report TI 03/00, TU-Darmstadt, 2000.
8. D. Hühnlein. Faster generation of NICE-Schnorr signatures. in preparation, 2000.
9. D. Hühnlein, M.J. Jacobson, Jr., S. Paulus, and T. Takagi. A cryptosystem based on non-maximal imaginary quadratic orders with fast decryption. In *Advances in Cryptology - EUROCRYPT '98*, volume 1403 of *LNCS*, pages 294–307, 1998.
10. D. Hühnlein, M.J. Jacobson, and D. Weber. Towards practical non-interactive public key cryptosystems using non-maximal imaginary quadratic orders. CORR technical report, www.cacr.math.uwaterloo.ca, to appear.
11. D. Hühnlein and J. Merkle. An efficient NICE-Schnorr-type signature scheme. In *Proceedings of Public Key Cryptography 2000*, Springer, volume 1751 of *LNCS*, pages 14–27, 2000.
12. D. Hühnlein and T. Takagi. Reducing logarithms in totally non-maximal imaginary quadratic orders to logarithms in finite fields. In *Advances in Cryptology - ASIACRYPT '99*, volume 1716 of *LNCS*, pages 219–231, 1999.
13. M.J. Jacobson, Jr. Computing discrete logarithms in quadratic orders. To appear *Journal of Cryptology*, 1999.

14. M.J. Jacobson, Jr. *Subexponential Class Group Computation in Quadratic Orders*. PhD thesis, Technische Universität Darmstadt, Darmstadt, Germany, 1999.
15. S. Lang. *Algebraic number theory*. Springer, Berlin, 2nd edition, 1991. ISBN 3-540-94225-4.
16. A.K. Lenstra, E. Verheul. Selecting Cryptographic Key Sizes. In *Proceedings of Public Key Cryptography 2000*, Springer, volume 1751 of *LNCS*, pages 446–465, 2000.
17. C.H. Lim and P.J. Lee. Modified Maurer-Yacobi's scheme and its applications. In *Proceedings of Auscrypt'92*, LNCS, pages 308–323, 1992.
18. M. Maurer and D. Kügler. A note on the weakness of the Maurer-Yacobi squaring method. Technical report, TI 15/99, TU Darmstadt, 1999.
19. U. Maurer and Y. Yacobi. Non-interactive public-key cryptography. In *Advances in Cryptology - EUROCRYPT'91*, volume 547 of *LNCS*, pages 498–507, 1991.
20. U. Maurer and Y. Yacobi. A remark on a non-interactive public-key distribution system. In *Advances in Cryptology - EUROCRYPT'92*, volume 658 of *LNCS*, pages 458–460, 1993.
21. U. Maurer and Y. Yacobi. A non-interactive public-key distribution system. *Design Codes and Cryptography*, 9:305–316, 1996.
22. R. Rivest. The MD5 message-digest algorithm, 1992. RFC1321, Internet Activities Board, Internet Engineering Task Force.
23. A. Shamir. Identity based cryptosystems and signature schemes. In *Advances in Cryptology - CRYPTO '84*, volume 196 of *LNCS*, pages 47–53, 1985.
24. D. Weber and T. Denny. The solution of McCurley's discrete log challenge. In *Advances in Cryptology - CRYPTO '98*, volume 1462 of *LNCS*, pages 56–60, 1998.

On the Implementation of Cryptosystems Based on Real Quadratic Number Fields (Extended Abstract)

Detlef Hühnlein¹ and Sachar Paulus²

¹ Secunet Security Networks AG, Germany
huehnlein@secunet.de

² SAP AG, Germany
sachar.paulus@t-online.com

Abstract. Cryptosystems based on the discrete logarithm problem in the infrastructure of a real quadratic number field [7,19,2] are very interesting from a theoretical point of view, because this problem is known to be at least as hard as, and when considering today's algorithms – as in [11] – much harder than, factoring integers. However it seems that the cryptosystems sketched in [2] have not been implemented yet and consequently it is hard to evaluate the practical relevance of these systems. Furthermore as [2] lacks any proofs regarding the involved approximation precisions, it was not clear whether the second communication round, as required in [7,19], really could be avoided without substantial slowdown. In this work we will prove a bound for the necessary approximation precision of an exponentiation using quadratic numbers in power product representation and show that the precision given in [2] can be lowered considerably. As the highly space consuming power products can not be applied in environments with limited RAM, we will propose a simple (CRIAD¹-) arithmetic which entirely avoids these power products. Beside the obvious savings in terms of space this method is also about 30% faster. Furthermore one may apply more sophisticated exponentiation techniques, which finally result in a ten-fold speedup compared to [2].

1 Introduction

Unlike for imaginary quadratic orders, there is no polynomial time algorithm known, which decides whether two given ideals in a real quadratic order \mathcal{O}_Δ are equivalent, and consequently it is impossible to set up DL-based cryptosystems in real quadratic class groups $Cl(\Delta)$. However, as noted by Shanks [21], there is some infrastructure in the cycle of reduced principal ideals, which resembles an abelian group. Buchmann, Williams and Scheidler [7,18,19] showed how to construct a Diffie-Hellman-like key agreement procedure using this infrastructure. They (essentially) represent a principal ideal \mathfrak{A} by a pair (\mathfrak{a}, a) , where $\mathfrak{a} = \gamma\mathfrak{A}$ is

¹ CRIAD is an abbreviation for Close Reduced Ideal and Approximated relative Distance

the reduced ideal *closest* to \mathfrak{A} and a is a rational approximation to the distance $\log(\gamma)$ between \mathfrak{a} and \mathfrak{A} . The major problem with their approach is that, because it is (in general) impossible to find the closest reduced ideal when using a fixed approximation precision, they compute a pair of very close reduced ideals, i.e. the left and right neighbour, and uniquely determine the common key in a *second communication round*. Because this additional round is necessary in their setup, it is impossible to construct more advanced – e.g. signature- – protocols. In [2] it was proposed to use the exact relative generator $\gamma \in \mathbb{Q}(\sqrt{\Delta})$ in power product representation [6] and only compute a rational approximation of its logarithm in the very end. The claimed that it is possible to avoid the second communication round and hence implement advanced cryptographic protocols (c.f. [3]). However as [2] lacks any proofs regarding the involved approximation precisions it was not clear whether the second communication round, as required in [7,19], can be avoided at all, and if, without substantially decreasing the efficiency.

In this work we will prove a bound for the necessary approximation precision of an exponentiation using quadratic numbers in power product representation and show that the precision given in [2] may be lowered considerably. For a discriminant with 1024 bit and 160 bit secret exponents in a Diffie-Hellman key-agreement protocol we will see that a precision of $512 + 2 + 160 + 3 = 677$ bits is sufficient, where [2] suggest a precision of $1024 + 6 \cdot 160 + 6 = 1990$ bits for this scenario. As in more sophisticated cryptographic protocols it is necessary to have, not only an exponentiation but also, a multiplication (and possibly inversion) routine available, we will introduce the so called CRIAD-multiplication, which is (essentially) associative (see Corollary 1). Using CRIADmult one is able to construct a binary exponentiation routine which requires a precision of $512 + 2 + 2 \cdot 160 + 2 = 836$ bits in the above scenario, but *entirely avoids power products* and solely uses floating point numbers for the logarithms. Note that this is very important in environments with restricted RAM. This binary exponentiation variant (CRIADexp Algorithm 3) with a precision of 836 bits already yields a running time, which is about 30% faster than the exponentiation variant using power products and an approximation precision of 677 bits. Another important feature of this approach is that one may use CRIADmult to implement more sophisticated exponentiation routines, which finally result in a ten-fold speedup compared to [2].

The paper is organized as follows: In Section 2 we will recall the necessary basics concerning the infrastructure of the principal class in a real quadratic number field. In Section 3 we carry together the necessary definitions concerning rational approximations of real numbers. In Section 4 we will recall the representation of principal ideals from [2] and prove lower bounds for the required approximation precision to provide a unique representation (see Proposition 2), which is necessary to avoid the second communication round, as required in [7,19]. In Section 5 we will show what precision is necessary to implement an exponentiation technique using power products. In Section 6 we will introduce the CRIAD-arithmetic which is necessary to implement more sophisticated protocols and allows to avoid the application of power products. In Section 7 we will

use this basic arithmetic to derive more sophisticated exponentiation techniques. Due to space restrictions we will only give the results and refer to [12] for a detailed presentation and proofs. We will conclude this work by providing timings of a first implementation in Section 8. In the full paper [12] we will also provide a complexity analysis and the treatment of more sophisticated cryptographic – e.g. signature – protocols.

2 The Infrastructure of the Principal Class in Real Quadratic Number Fields

In this section we will recall the very basic notions of the infrastructure of the principal class in real quadratic number fields needed in the sequel and refer to [8, 10, 1, 13] for a complete reference.

Let $\mathbb{Q}(\sqrt{\Delta})$ be the real quadratic number field of discriminant $\Delta > 0$ and \mathcal{O}_Δ be its ring of integers. We denote \mathcal{O}_Δ -ideals by gothic letters $\mathfrak{a}, \mathfrak{b}, \dots, \mathfrak{A}, \mathfrak{B}, \dots$. An important invariant of the number field $\mathbb{Q}(\sqrt{\Delta})$ is the regulator $\mathcal{R}_\Delta = \log \varepsilon$, where ε is the smallest unit larger than one. It is well known (see e.g. [20]), that the computation of \mathcal{R}_Δ is at least as hard as factoring Δ .

Two \mathcal{O}_Δ -ideals \mathfrak{a} and \mathfrak{b} are called *equivalent* if there is a $\gamma \in \mathbb{Q}(\sqrt{\Delta})$ such that $\mathfrak{a}\gamma = \mathfrak{b}$. γ is called a *relative generator* of \mathfrak{b} w.r.t. \mathfrak{a} . γ is unique up to multiplication by units. Equivalence of ideals is an equivalence relation. The equivalence classes are called *ideal classes*. If $\mathfrak{a} = \mathcal{O}_\Delta$ then $\mathfrak{b} = \gamma\mathcal{O}_\Delta$ is called a *principal ideal* and γ is simply called a *generator* of \mathfrak{b} . The set of principal ideals is denoted by \mathcal{P}_Δ and forms an infinite abelian subgroup of \mathcal{I}_Δ . The factor group $Cl(\Delta) = \mathcal{I}_\Delta/\mathcal{P}_\Delta$ is a finite abelian group and is called the *ideal class group* of $\mathbb{Q}(\sqrt{\Delta})$.

For two equivalent ideals \mathfrak{a} and $\mathfrak{b} = \gamma\mathfrak{a}$ we define the *distance* between \mathfrak{a} and \mathfrak{b} by

$$\delta(\mathfrak{a}, \mathfrak{b}) = \frac{1}{2} \log \left| \frac{\gamma}{\bar{\gamma}} \right| \pmod{\mathcal{R}_\Delta}, \quad (1)$$

where $\bar{\gamma} = (x - y\sqrt{\Delta})/z$ denotes the (real) conjugate of $\gamma = (x + y\sqrt{\Delta})/z$. If $\mathfrak{a} = \mathcal{O}_\Delta$, we simply write $\delta(\mathfrak{b})$ instead of $\delta(\mathcal{O}_\Delta, \mathfrak{b})$.

Given an ideal \mathfrak{A} we denote by $\rho()$ (see e.g. [13, REDUCE_REAL, Algorithm 2.6]) the reduction operator, which computes an equivalent reduced ideal $\mathfrak{a} = \rho(\mathfrak{A})$. We denote $(n \geq 1)$ successive applications of $\rho()$ by $\rho^n()$. If \mathfrak{a} is a reduced ideal then $\rho^n(\mathfrak{a})$ is also reduced and there is some $l \in \mathbb{Z}_{>1}$, such that $\mathfrak{a} = \rho^l(\mathfrak{a})$. In [22] it is shown that, for arbitrary Δ , the smallest such l may be as large as $O(\sqrt{\Delta} \log \log \Delta)$.

Let $\mathfrak{a} = (a, b)$ be a reduced ideal. Then one may use [13, Algorithm 2.11] to compute the *right neighbour* $\mathfrak{a}_+ = (a_+, b_+) = \rho(\mathfrak{a})$, where $\delta(\mathfrak{a}, \mathfrak{a}_+) = \frac{1}{2} \log \left| \frac{b_+ + \sqrt{\Delta}}{b_+ - \sqrt{\Delta}} \right|$ and [13, Algorithm 2.12] respectively to compute the *left neighbour* $\mathfrak{a}_- = (a_-, b_-) = \rho^{-1}(\mathfrak{a})$, where $\delta(\mathfrak{a}, \mathfrak{a}_-) = -\frac{1}{2} \log \left| \frac{b_- + \sqrt{\Delta}}{b_- - \sqrt{\Delta}} \right|$ of \mathfrak{a} .

It is easy to see that traveling around a complete circle and obtaining $\rho^l(\mathfrak{a}) = \mathfrak{a}$ and thus $\mathfrak{a}(\epsilon) = \mathfrak{a}$ yields

$$\delta(\rho^l(\mathfrak{a}), \mathfrak{a}) = \frac{1}{2} \log \left| \frac{\epsilon}{\bar{\epsilon}} \right| = \frac{1}{2} \log \left| \frac{\epsilon^2}{N(\epsilon)} \right| = \frac{1}{2} \log(\epsilon^2) = \log \epsilon = \mathcal{R}_\Delta.$$

Instead of this naive strategy, which is obviously only applicable for very small Δ , Shanks [21] made use of the *infrastructure* of the principal class of a real quadratic order to compute \mathcal{R}_Δ . In the following we will recall the most important properties of this infrastructure.

In [15] it was shown that

$$\log(1/\sqrt{\Delta} + 1) < |\delta(\mathfrak{a}, \mathfrak{a}_+)| < \log \sqrt{\Delta}, \quad (2)$$

and for three consecutive ideals $\mathfrak{a}, \mathfrak{a}_+, \mathfrak{a}_{++}$ we have

$$|\delta(\mathfrak{a}, \mathfrak{a}_{++})| > \log 2. \quad (3)$$

Moreover, it is immediate from the definition that for three equivalent ideals $\mathfrak{a}, \mathfrak{b}, \mathfrak{c}$ we have

$$\delta(\mathfrak{a}, \mathfrak{b}) + \delta(\mathfrak{b}, \mathfrak{c}) \equiv \delta(\mathfrak{a}, \mathfrak{c}) \pmod{\mathcal{R}_\Delta} \quad (4)$$

and for two pairs $(\mathfrak{a}, \mathfrak{b})$ and $(\mathfrak{c}, \mathfrak{d})$ of principal ideals

$$\delta(\mathfrak{ac}, \mathfrak{bd}) \equiv \delta(\mathfrak{a}, \mathfrak{c}) + \delta(\mathfrak{b}, \mathfrak{d}) \pmod{\mathcal{R}_\Delta}. \quad (5)$$

The last assertions are not surprising, as the set of (invertible) principal ideals \mathcal{P}_Δ forms a group under multiplication. On the other hand it is easy to see that the subset of *reduced* principal ideals, together with some combination of multiplication and reduction, does not form a group, because such an operation is either not associative or not closed.

However it is possible to show that the operation \odot , defined by multiplication followed by reduction, is “close to” being a group operation:

Proposition 1. *Let $\mathfrak{a}, \mathfrak{b}$ be reduced ideals and $\mathfrak{c} = \mathfrak{a} \odot \mathfrak{b} = \rho(\mathfrak{ab})$. Then*

$$|\delta(\mathfrak{c}, \mathfrak{ab})| = |\delta(\mathfrak{c}) - (\delta(\mathfrak{a}) + \delta(\mathfrak{b}))| < 2 \log \Delta. \quad (6)$$

Proof. See [8, Proposition 5.8.4].

We will see that this deficiency w.r.t. a group operation can be repaired, without applying the space consuming power product representation from [6]. In the multiplication procedure for principal ideals in CRIAD-representation (CRIADmult, Algorithm 2) we will additionally compute rational approximations for distances, which allow to correct the “error” introduced by reduction. We will see in Corollary 1, that this strategy makes the operation CRIADmult (essentially) associative, if one uses a sufficiently high approximation precision.

3 Rational Approximations of Real Numbers

The distances between equivalent ideals, as defined in (1), are of the form $1/2 \log |\gamma/\bar{\gamma}|$, for $\gamma \in \mathbb{Q}(\sqrt{\Delta})^*$, and hence in general irrational numbers. As the most practical way to handle these distances seems to be the computation of – sufficiently accurate – rational approximations, we will follow [5,17] and carry together the necessary definitions and properties of *floating point numbers*.

Let $r \in \mathbb{R}$, $r \neq 0$. Then we define $b(r) = \lfloor \log_2 |r| \rfloor + 1$ and $b(0) = 0$.

Definition 1. A floating point number is a pair $f = (m, e)$, where $m, e \in \mathbb{Z}$, $m \neq 0$ or $m = 0$ and $e = 0$. m is called the mantissa and e is called the exponent of f . $f = (m, e)$ represents the rational number $q = m \cdot 2^{e-b(m)}$.

Definition 2. Let $r \in \mathbb{R}$ and $k \in \mathbb{Z}, j \in \mathbb{Q}_{>0}$.

1. An absolute k -approximation for r is a floating point number $f = (m, e)$, such that $|f - r| < 2^{-k}$ and $e \geq b(m) - k - 1$.
2. An absolute (j, k) -approximation for r is a floating point number $f = (m, e)$, such that $|f - r| < \frac{j}{2^k}$ and $e \geq b(m) - k + \lceil \log_2(j) \rceil - 1$.

Remark 1. The latter definition is necessary to consider the round-off-error in some computation more closely. Here we will shortly relate a (j, k) -approximation to an l -approximation.

Let $f = (m, e)$, where $e \geq b(m) - k + \lceil \log_2(j) \rceil - 1$, be an absolute (j, k) -approximation for $r \in \mathbb{R}$. Then $|f - r| < \frac{j}{2^k} = 2^{\log_2(\frac{j}{2^k})} = 2^{\log_2(j) - k}$ and one immediately sees that f is precisely an absolute l -approximation for $l = k - \lceil \log_2(j) \rceil$. On the other side it is clear from the definition that a $(1, k)$ -approximation is precisely a k -approximation.

We use Markus Maurer's functions from the `xbigfloat`-class of LiDIA [16] to implement the necessary floating point arithmetic. A theoretical treatment of these functions may be found in [17]. Besides addition, subtraction and the comparison of floating point numbers we will also need the following two functions:

- `Trunc(f, k)`
denotes the LiDIA-function `Truncate(f, k)` and returns the “ k significant bits” of a floating point number $f = (m, e) = m \cdot 2^{e-b(m)}$ by deleting the last $b(m) - k$ last bits of the mantissa m . To prove a bound for the necessary precision to make our proposed CRIAD-representation unique, we will make use of [17, Lemma 2.3.1], which states that given a k -approximation $f = (m, e)$ of a real number r , $f' = \text{Trunc}(f, k + e)$ is a $k - 1$ -approximation of r .
- `qlog(x, y, k)`
denotes the LiDIA-function `a.absolute.Ln.approximation(k)` and returns on input of a number $\gamma = x + y\sqrt{\Delta}$ a k -approximation of Lenstra's [15] logarithm $\text{Log}(\gamma) = 1/2 \log |\gamma/\bar{\gamma}|$. A thorough description of this function may be found in [17, Section 6.1.4].

4 Bounds for the Uniqueness of the CRIAD-Representation

In this section we introduce the CRIAD-representation of principal ideals, as sketched in [2]. After a formal specification of the required properties we will derive bounds for the involved precision to make this representation unique.

To make the line of thought leading to this important representation more transparent, we will start with defining a RIAD-representation, where we do not require that the reduced ideal in this representation is close to the represented ideal.

Definition 3. Let \mathfrak{A} be a (fractional) principal \mathcal{O}_Δ -ideal, $f = (m, e)$ a floating point number and $k \in \mathbb{Z}, j \in \mathbb{Q}_{>0}$.

A (j, k) -RIAD-representation of \mathfrak{A} is a pair (\mathfrak{a}, f) , where \mathfrak{a} is an arbitrary reduced principal ideal and f is an absolute (j, k) -approximation for the distance $\delta(\mathfrak{A}, \mathfrak{a})$ between \mathfrak{A} and \mathfrak{a} . A $(1, k)$ -RIAD-representation is simply called k -RIAD-representation.

Given \mathfrak{a} , not necessarily reduced, principal ideal \mathfrak{A} in standard representation it is easy to compute a k -RIAD-representation (\mathfrak{a}, f) for it. The procedure $(\mathfrak{a}, f) = \text{Std2RIAD}(\mathfrak{A}, k)$ uses the standard LiDIA-routine $(\mathfrak{a}, \gamma) = \text{REDUCE}(\mathfrak{A})$ (see e.g. [13, REDUCE_REAL, Algorithm 2.6]), which computes $\mathfrak{a} = \rho(\mathfrak{A})$ and the relative generator $\gamma = (x + y\sqrt{\Delta})/z = \mathfrak{A}/\mathfrak{a}$, and computes $f = -\mathbf{qlog}(x, y, k)$.

As the reduced ideal \mathfrak{a} in this representation will, for example, be used to derive a common key in a Diffie-Hellman key agreement procedure, it is especially important to guarantee that both communication partners end up with the same reduced ideal \mathfrak{a} , representing $\mathfrak{A} = \mathfrak{g}^{ab}$, while performing entirely different computations.

As there are no further requirements for the reduced ideal \mathfrak{a} in the RIAD-representation (\mathfrak{a}, f) of a principal ideal \mathfrak{A} and there are $c = O(\sqrt{\Delta} \log \log \Delta)$ reduced ideals in the principal cycle, there are obviously c different RIAD-representations for \mathfrak{A} . Among all these RIAD-representations for \mathfrak{A} we will now elect the CRIAD-representation, which will be shown to be uniquely determined if the involved precision is sufficiently high. If the precision would be too low, such that there would be two valid CRIAD-representations (\mathfrak{a}_i, f_i) , $i \in \{1, 2\}$, $\mathfrak{a}_1 \neq \mathfrak{a}_2$, for an ideal $\mathfrak{A} = \mathfrak{g}^{ab}$, then a key agreement procedure would entirely fail to work, or would need to be “repaired” using a second communication round, as proposed in [7, 19].

Suppose for a moment, that the distances could be determined exactly. Then one could simply define the unique representative for \mathfrak{A} to be the reduced ideal \mathfrak{a} with the (in absolute value) smallest distance and possibly – if \mathfrak{A} is *precisely* inbetween two reduced ideals – positive distance. In this case it is clear that \mathfrak{a} is uniquely determined.

However, since we are dealing with rational approximations, i.e. we only have $k < \infty$ many correct bits of the distances at our disposal, some more considerations are necessary to make the reduced ideal in the CRIAD-representation unique.

Suppose, that all computations were performed with a sufficiently high precision k , that we can guarantee, that the reduced ideal \mathfrak{a} in the RIAD-representation (\mathfrak{a}, f) is either the left or right neighbour of \mathfrak{A} and recall that the function $\text{Trunc}(g, l)$ returns (only) the l significant bits of a floating point number g .

Let (\mathfrak{a}_1, f_1) , with $f_1 = (m_1, e_1)$, and (\mathfrak{a}_2, f_2) , with $f_2 = (m_2, e_2)$, be two candidates for the CRIAD-representation. If $|\text{Trunc}(f_1, k + e_1)| < |\text{Trunc}(f_2, k + e_2)|$, then the decision is easy and we choose (\mathfrak{a}_1, f_1) to be the unique CRIAD-representation.

In the worst case \mathfrak{A} is – in the scope of the fixed approximation precision – precisely inbetween the two reduced ideals \mathfrak{a}_1 and \mathfrak{a}_2 . Then the RIAD-representations (\mathfrak{a}_1, f_1) and (\mathfrak{a}_2, f_2) have the following properties:

For the distances we have

$$|\delta(\mathfrak{A}, \mathfrak{a}_1) - f_1| < 2^{-k} \text{ and } |\delta(\mathfrak{A}, \mathfrak{a}_2) - f_2| < 2^{-k}, \quad (7)$$

since we have k -RIAD-representations and

$$\text{Trunc}(f_1, k + e_1) = -\text{Trunc}(f_2, k + e_2), \quad (8)$$

since \mathfrak{A} is – in the scope of the fixed approximation precision – precisely inbetween \mathfrak{a}_1 and \mathfrak{a}_2 .

We assume w.l.o.g. that $f_1 > 0$ and choose (\mathfrak{a}_1, f_1) to be the unique CRIAD-representation for \mathfrak{A} , even if the precise distances may satisfy

$$|\delta(\mathfrak{A}, \mathfrak{a}_1)| > |\delta(\mathfrak{A}, \mathfrak{a}_2)|.$$

Thus the reduced ideal \mathfrak{a} in the CRIAD-representation (\mathfrak{a}, f) is not necessarily the reduced ideal closest to \mathfrak{A} , but nevertheless *uniquely determined*, if the approximation precision, as part of the system parameters, is sufficiently high.

Now we will bring the above vague ideas in a more formal shape, such that we will be able to determine the necessary precision in order to guarantee that the reduced ideal \mathfrak{a} in a CRIAD-representation is uniquely determined.

Definition 4. Let $k \in \mathbb{Z}$, $j \in \mathbb{Q}_{>0}$ and $l = k - \lceil \log_2(j) \rceil$. Then a (j, k) -CRIAD-representation of \mathfrak{A} is defined to be a (j, k) -RIAD-representation (\mathfrak{a}, f) , where $f = (m, e)$, of \mathfrak{A} , satisfying the following properties:

1. $|\text{Trunc}(f, l + e)| \leq |\text{Trunc}(f', l + e')|$ for all (j, k) -RIAD-representations (\mathfrak{a}', f') , where $f' = (m', e')$, of \mathfrak{A} and
2. if (\mathfrak{a}_1, f_1) and (\mathfrak{a}_2, f_2) are two (j, k) -RIAD-representations, which satisfy (1.), where $f_1 > 0$ and $f_2 < 0$, then $(\mathfrak{a}, f) = (\mathfrak{a}_1, f_1)$.

If \mathfrak{A} is reduced, then we call $(\mathfrak{A}, 0)$ a $(0, k)$ -CRIAD-representation for any $k \in \mathbb{Z}$. A $(1, k)$ -CRIAD-representation is simply called a k -CRIAD-representation.

Definition 5. A (j, k) -CRIAD-representation (\mathfrak{a}, f) is called *unique*, if there is no $l = k - \lceil \log_2(j) \rceil$ -CRIAD-representation (\mathfrak{a}', f') , $\mathfrak{a}' \neq \mathfrak{a}$, for a given ideal \mathfrak{A} , which satisfies (1.) and possibly (2.) in above definition.

It remains to determine a bound for the precision to make such a CRIAD-representation unique. For this purpose we will proceed in two steps. In Lemma 1 we will present a different formulation of the uniqueness-problem for CRIAD-representations. We will show that this representation is unique if in a certain real, open, interval of width $2^{-k+\lceil\log_2(j)\rceil+2}$ there is only one reduced ideal. In Proposition 2 we will derive a bound such that in an interval of said width there can be only one reduced ideal.

Lemma 1. *Let (\mathfrak{a}_1, f_1) , where $f_1 = (m_1, e_1)$, be a (j, k) -CRIAD-representation of a principal ideal \mathfrak{A} , $l = k - \lceil\log_2(j)\rceil$ and $f = \text{Trunc}(f_1, l + e_1)$. This CRIAD-representation is unique, if there is no reduced ideal $\mathfrak{a}_2 \neq \mathfrak{a}_1$, such that*

$$\delta(\mathfrak{a}_2) \in]\delta(\mathfrak{A}) - f - 2^{-l+1}, \delta(\mathfrak{A}) - f + 2^{-l+1}[.$$

Proof. Let $l = k - \lceil\log_2(j)\rceil$. Considering the above definition, we see that the (j, k) -CRIAD-representation (\mathfrak{a}_1, f_1) , with $f_1 = (m_1, e_1)$, for a principal ideal \mathfrak{A} is not unique if there is (at least) one other (j, k) -CRIAD-representation (\mathfrak{a}_2, f_2) , with $f_2 = (m_2, e_2)$, for \mathfrak{A} , such that $\mathfrak{a}_2 \neq \mathfrak{a}_1$ and $f = \text{Trunc}(f_1, l + e_1) = \text{Trunc}(f_2, l + e_2)$.

Considering the involved distances, we have

$$|\delta(\mathfrak{A}, \mathfrak{a}_i) - f_i| < j/2^k \leq 2^{-l}, \quad i \in \{1, 2\},$$

as (\mathfrak{a}_i, f_i) are (j, k) -CRIAD-representations.

By [17, Lemma 2.3.1] we loose one bit of precision by computing

$$f = \text{Trunc}(f_1, l + e_1) = \text{Trunc}(f_2, l + e_2).$$

I.e. as f_i , $i \in \{1, 2\}$, are absolute l -approximations for $\delta(\mathfrak{A}, \mathfrak{a}_i)$ we know that f is an absolute $l - 1$ -approximation for $\delta(\mathfrak{A}, \mathfrak{a}_i)$ and we obtain

$$|\delta(\mathfrak{A}, \mathfrak{a}_i) - f| = |\delta(\mathfrak{A}) - f - \delta(\mathfrak{a}_i)| < 2^{-l+1}, \quad i \in \{1, 2\}.$$

This shows that non-uniqueness occurs, if there is some reduced ideal $\mathfrak{a}_2 \neq \mathfrak{a}_1$, such that

$$\delta(\mathfrak{a}_2) \in]\delta(\mathfrak{A}) - f - 2^{-l+1}, \delta(\mathfrak{A}) - f + 2^{-l+1}[. \quad \square$$

Looking back to our original argumentation, Lemma 1 shows that non-uniqueness occurs, if there is a reduced ideal \mathfrak{a}' , such that the CRIAD-representation (\mathfrak{a}', f') satisfies all requirements in the definition, but \mathfrak{a}' is *not the direct left or right neighbour* of \mathfrak{A} .

To derive a bound for the uniqueness of the CRIAD-representation, it is – by Lemma 1 – sufficient to investigate, whether in a real, open interval of width $2^{-k+\lceil\log_2(j)\rceil+2}$ there may be two (or more) reduced ideals.

Proposition 2. *Let (\mathfrak{a}, f) be a (j, k) -CRIAD-representation of a principal \mathcal{O}_Δ -ideal \mathfrak{A} , $l = k - \lceil\log_2(j)\rceil$ and*

$$\chi(\Delta) = -\log_2 \left(\log \left(1/\sqrt{\Delta} + 1 \right) \right) + 2. \quad (9)$$

Then the (j, k) -CRIAD-representation (\mathfrak{a}, f) of \mathfrak{A} is unique, if $l \geq \chi(\Delta)$.

Proof. Let $l = k - \lceil \log_2(j) \rceil$. Then – by Lemma 1 – it is sufficient to explore, whether two neighbouring, reduced ideals $\mathfrak{a}, \mathfrak{a}_+$ may lie in an open interval of width 2^{-l+2} . By (2) we have

$$|\delta(\mathfrak{a}, \mathfrak{a}_+)| > \log(1/\sqrt{\Delta} + 1). \quad (10)$$

We have uniqueness, if it is impossible that two neighbouring ideals lie in the interval of width 2^{-l+2} . By (10) we have $2^{-l+2} \leq \log(1/\sqrt{\Delta} + 1) < |\delta(\mathfrak{a}, \mathfrak{a}_+)|$ and therefore

$$l \geq -\log_2(\log(1/\sqrt{\Delta} + 1)) + 2 = \chi(\Delta). \quad \square$$

Remark 2. During the execution of a cryptographic protocol one needs to take care that one remains above this minimum precision.

For $x > 1$ we have $\log(1/x + 1) > 1/(x + 1)$ and one obtains the bound

$$\chi(\Delta) < \log_2(\sqrt{\Delta} + 1) + 2. \quad (11)$$

Thus it is sufficient, that – at the end of any cryptographic protocol – about $\log_2(\Delta)/2$ correct bits of the distances are at our disposal. Note that the cursory “analysis” in [2] suggests a minimum precision of $\log_2(\Delta)$ bits.

5 CRIAD-Exponentiation Using Power Products

In this section we will show what precision is sufficient in an exponentiation routine for ideals in CRIAD-representation using power products, as in [2]. As above, our analysis will reveal that the precision bounds given in [2] are way too high.

We will use the LiDIA-function $(\mathfrak{b}, d) = \text{CLOSE}(\mathfrak{a}, t, k)$ which on input of a reduced ideal \mathfrak{a} , a rational number t and an approximation precision k will return a reduced ideal \mathfrak{b} such that $\delta(\mathfrak{b})$ is – with respect to k – close to $\delta(\mathfrak{a}) + t$ and a k -approximation d to $\text{Log}(\mathfrak{a}/\mathfrak{b})$. A detailed description of this function can be found in [17, Section 8.4]. Note that the functionality of **CLOSE** is equal to the functionality of the procedure **TARGET** in [2].

Let $n \in \mathbb{Z}_{>0}$ and $l = \lceil \log_2(n) \rceil$. Then (n_l, \dots, n_0) is the binary expansion of $n = \sum_{i=0}^l n_i 2^i$, where $n_i \in \{0, 1\}$ for $0 \leq i \leq l-1$, $n_l = 1$.

Proof. Recall that in any call $(\mathfrak{a}, \alpha) = \text{REDUCE}(\mathfrak{A})$ we have $\mathfrak{a} = \mathfrak{A}/\alpha$.

Thus we obtain at the end of the **for**-loop $\mathfrak{a}^n = \mathfrak{h}\gamma$ and compute the floating point number t such that $\delta(\mathfrak{h}) + t$ is close to $\delta(\mathfrak{A}^n)$. Therefore the correctness, disregarding the approximation precision, follows from the correctness of **CLOSE** [17, Section 8.4].

It remains to show the correctness of the J -value. Let $\theta = \mathfrak{A}^n/\mathfrak{h}$. Then we have $|t - \text{Log}(\theta)| < nj2^{-k} + 2^{-(k+z)} \leq (2^{l+1} - 1)j2^{-k} + 2^{-(k+z)}$ and at the very end $|\mathfrak{A}^n - (\delta(\mathfrak{h}) + h)| < (2^{l+1} - 1)j2^{-k} + 3 \cdot 2^{-(k+z)}$ which shows that $J = (2^{l+1} - 1)j + 3 \cdot 2^{-z}$ is correct. \square

Algorithm 1 CRIADexpPP

Require: The (j, k) -CRIAD-representation (\mathbf{a}, f) of a principal \mathcal{O}_Δ -ideal \mathfrak{A} , the exponent $n = (n_l, \dots, n_0) \in \mathbb{Z}_{>0}$, the final precision k and the additional precision $z \geq 0$.

Ensure: The (J, k) -CRIAD-representation (\mathbf{h}, g) of \mathfrak{A}^n , where $J = (2^{l+1} - 1)j + 3 \cdot 2^{-z}$.

```

 $\gamma = 1$ 
 $\mathbf{h} = \mathbf{a}$ 
for  $i = l - 1$  downto  $0$  do
   $(\mathbf{h}, \alpha) \leftarrow \text{REDUCE}(\mathbf{h}^2)$ 
   $\gamma \leftarrow \gamma^2 \alpha$ 
  if  $n_i = 1$  then
     $(\mathbf{h}, \alpha) \leftarrow \text{REDUCE}(\mathbf{h} \cdot \mathbf{a})$ 
     $\gamma \leftarrow \gamma \alpha$ 
  end if
end for
 $t \leftarrow f \cdot n - \text{qlog}(\gamma, k + z)$ 
 $(\mathbf{h}, h) \leftarrow \text{CLOSE}(\mathbf{h}, t, k + z)$ 
 $h \leftarrow t - h$ 
return  $(\mathbf{h}, h)$ 

```

To allow a fair comparison of the different exponentiation strategies we need to consider their behaviour within some cryptographic protocol. We will only give bounds for the Diffie-Hellman key-agreement-protocol here and treat more sophisticated protocols in the final paper [12].

Proposition 3. *Let $a, b < 2^{l+1}$ be the secret exponents in a Diffie-Hellman key-agreement protocol using CRIADexpPP and a reduced ideal as common base. Then it is sufficient if both partners use $k \geq \chi(\Delta)$ and $z \geq l + 3$.*

Proof. As both partners start with a reduced ideal, i.e. a $(0, k)$ -CRIAD-representation, they obtain a (j_1, k) CRIAD-representation with the first exponentiation, where $j_1 = 3 \cdot 2^{-z}$. The second exponentiation yields a (j_2, k) -representation, where $j_2 = (2^{l+1} - 1)j_1 + 3 \cdot 2^{-z} = 3 \cdot 2^{l+1-z} < 2^{l+3-z}$. This is a k -approximation, if $z \geq l + 3$. \square

This proposition explains the required precision of $512 + 2 + 160 + 3 = 677$ bits, for 1024 bit Δ and 160 bit exponents, stated in the introduction.

6 CRIAD-Arithmetic without Power Products

Regardless of the applied exponentiation technique it is necessary to have the procedure CRIADmult – and possibly CRIADinv – available to implement more sophisticated – e.g. signature – protocols. Therefore we will develop this basic

arithmetic for principal ideals in CRIAD-representation. We will show in Corollary 1, that CRIADmult is (essentially) a group operation. Thus one may construct exponentiation techniques based on this procedure which do not require the power product representation and consequently can be applied in environments with limited RAM.

In our presentation of CRIADmult we will need a procedure $(\mathfrak{b}, g) = \text{RIAD2CRIAD}((\mathfrak{a}, f), z)$ which uses right- or leftsteps to convert a given (j, k) -RIAD-representation approximation into the $(j+1, k)$ -CRIAD-representation. One may² use the procedure LOCAL_CLOSE [17, Section 8.2] for this purpose.

Algorithm 2 CRIADmult

Require: The (j_a, k) -CRIAD-representation (\mathfrak{a}, a) of a principal ideal \mathfrak{A} , the (j_b, k) -CRIAD-representation of a principal ideal \mathfrak{B} , the final precision k and an additional precision $z \in \mathbb{Z}_{\geq 0}$, where $k \geq \chi(\Delta) + \log_2(j_a + j_b + 2^{-z})$.

Ensure: The unique $(j_a + j_b + 2^{-z}, k)$ -CRIAD-representation (\mathfrak{c}, c) of $\mathfrak{A}\mathfrak{B}$.

```

 $p \leftarrow k + z + 1$ 
 $(\mathfrak{h}, h) \leftarrow \text{Std2RIAD}(\mathfrak{a} \cdot \mathfrak{b}, p)$ 
 $(\mathfrak{c}, c) \leftarrow \text{RIAD2CRIAD}((\mathfrak{h}, h + a + b), p)$ 
return $(\mathfrak{c}, c)$ 

```

Proof. The proof will appear in the full paper [12]. □

Now it is easy to see that this operation is (essentially) associative, provided that the approximation precision is chosen to be sufficiently large.

Corollary 1. *Let $z \geq 0$ and $(\mathfrak{a}, a), (\mathfrak{b}, b), (\mathfrak{c}, c)$ be the unique $(j_a, k), (j_b, k), (j_c, k)$ -CRIAD-representations for the principal ideals $\mathfrak{A}, \mathfrak{B}, \mathfrak{C}$, $J = j_a + j_b + j_c + 2^{-z+1}$, where $k \geq \chi(\Delta) + \lceil \log_2(J) \rceil$. Let $(\mathfrak{d}_1, d_1) = \text{CRIADmult}(\text{CRIADmult}((\mathfrak{a}, a), (\mathfrak{b}, b), z), (\mathfrak{c}, c), z)$, where $d_1 = (m_1, e_1)$ and $(\mathfrak{d}_2, d_2) = \text{CRIADmult}((\mathfrak{a}, a), \text{CRIADmult}((\mathfrak{b}, b), (\mathfrak{c}, c), z), z)$, where $d_2 = (m_2, e_2)$. Then $\mathfrak{d}_1 = \mathfrak{d}_2$ and $\text{Trunc}(d_1, k + e_1) = \text{Trunc}(d_2, k + e_2)$.*

Proof. See [12]. □

Remark 3. Since we have chosen Lenstra's distance $\text{Log}(\gamma) = 1/2 \log |\gamma/\bar{\gamma}|$, as proposed in [15], instead of Shanks' naive distance $\log(\gamma)$ [21], it is easy to see that the *inversion* of a principal ideal in CRIAD-representation is essentially free of cost and especially does not impose any round-off-errors. If $((a, b), f)$ is a (j, k) -CRIAD-representation of \mathfrak{A} , then $((a, -b), -f) = \text{CRIADinv}((a, b), f)$ is a

² Note that LOCAL_CLOSE makes use of (small) power products. Due to space restrictions we need to refer to the final paper [12] for our method RIAD2CRIAD , which avoids power products at the cost of a slightly higher internal precision

(j, k) -RIAD³-representation of \mathfrak{A}^{-1} . This fact will be used to construct signed digit exponentiation routines, which are slightly more efficient.

7 CRIAD-Exponentiation without Using Power Products

As CRIADmult essentially behaves like a group operation it is straightforward to construct – more sophisticated – exponentiation routines for principal ideals in CRIAD-representation.

Due to space restrictions we will only present the exponentiation routine based on the classical binary square and multiply strategy in detail here. For the more sophisticated exponentiation routines we will only present the results of the precision analysis; the corresponding proofs appear in the full paper [12].

Algorithm 3 CRIADexp

Require: The (j, k) -CRIAD-representation (\mathfrak{a}, f) of a principal \mathcal{O}_Δ -ideal \mathfrak{A} , the exponent $n = (n_l, \dots, n_0) \in \mathbb{Z}_{>0}$, the final precision k and the additional precision $z \geq 0$.

Ensure: The (J, k) -CRIAD-representation (\mathfrak{b}, g) of \mathfrak{A}^n , where $J = (2^{l+1} - 1)j + 2^{-z}(2^{l+1} - 2)$.

```

 $(\mathfrak{h}, h) \leftarrow (\mathfrak{a}, f)$ 
for  $i = l - 1$  downto  $0$  do
   $(\mathfrak{h}, h) \leftarrow \text{CRIADmult}((\mathfrak{h}, h), (\mathfrak{h}, h), k, z)$ 
  if  $n_i = 1$  then
     $(\mathfrak{h}, h) \leftarrow \text{CRIADmult}((\mathfrak{h}, h), (\mathfrak{a}, f), k, z)$ 
  end if
end for
return  $(\mathfrak{h}, h)$ 

```

Remark 4. It should be noted that the presented square and multiply strategy is the so called “left-to-right” variant. This is important, because it features *less error propagation* for reduced ideals than the “right-to-left” variant [14], while the number of group operations is the same. This is yet another point, where the precision stated in [2] can be easily improved.

Proposition 4. *Let $a, b < 2^{l+1}$ be the secret exponents in a Diffie-Hellman key-agreement protocol using CRIADexp and a reduced ideal as common base. Then it is sufficient if both partners use $k \geq \chi(\Delta)$ and $z \geq 2l + 2$.*

In a similar manner we obtain the following bounds for more sophisticated exponentiation techniques; for the proofs we need to refer to the final paper [12].

³ It should be noted that, if the ideal is *precisely* inbetween two reduced ideals, another right step might be necessary to obtain the CRIAD-representation

Proposition 5. *On input of a (j, k) -CRIAD-representation and the exponent n with $l = \lfloor \log_2(n) \rfloor$, the sliding m -bit window method (see e.g. [9]) produces a (J, k) -CRIAD-representation, where $J = 2^{l+m+1}j + 2^{l+m+1}2^{-z}$.*

Let $a, b < 2^{l+1}$ be the secret exponents in a Diffie-Hellman key-agreement protocol using CRIADexpwindow and a reduced ideal as common base. Then it is sufficient if both partners use $k \geq \chi(\Delta)$ and $z \geq 2(l + m) + 3$.

Proposition 6. *On input of a (j, k) -CRIAD-representation, the appropriate pre-computed values and the exponent n with $l = \lfloor \log_2(n) \rfloor$, the signed 2^m -digit version of the BGMW exponentiation method [4] produces a (J, k) -CRIAD-representation, where $J = 2^{l+m+2}j + 2^{l+m+3}2^{-z}$.*

Let $a, b < 2^{l+1}$ be the secret exponents in a Diffie-Hellman key-agreement protocol using CRIADexpBGMW and a reduced ideal as common base. Then it is sufficient if both partners use $k \geq \chi(\Delta)$ and $z \geq 2(l + m) + 6$.

If the first exponentiation is performed with CRIADexpBGMW and the second exponentiation is performed with CRIADexpwindow, which might be often used in practice, then it is sufficient if both partners use $k \geq \chi(\Delta)$ and $z \geq 2(l + m) + 5$.

8 Timings of a First Implementation

In this section we provide timings of a first implementation using the different exponentiation methods discussed in this work. For the sake of comparison, we also provide timings of an implementation of the procedure EXP as given in [2]. The timings in Table 1 are given in seconds and correspond to randomly chosen discriminants and exponents of the respective bit length on a Pentium II with 166 MHz using LiDIA 2.0 [16]. As precision we used the necessary precision for the Diffie-Hellman protocol as given in the Propositions 3, 4, 5, 6 and [2] respectively.

The results for 500 bit discriminants and 100 bit exponents, which are closest to real world requirements, indicate that an exponentiation with CRIADexp is more than twice as fast as the exponentiation routine EXP from [2] and can – using precomputation – be accelerated to obtain a more than ten times faster exponentiation. Moreover our approach without applying power products seems to save not only a considerable amount of space, but also up to 30% time. Because the relative speedup tends to increase with higher parameters one might conjecture that our method is also asymptotically preferable. This issue will be discussed in the full paper [12].

Acknowledgement

We would like to thank Renate Scheidler, Markus Maurer and Hugh C. Williams for fruitful discussions and for making us aware of mistakes in an earlier draft of this paper.

Table 1. Timings for different CRIAD-exponentiations

| Bitlength of Exponent | Bitlength of Discriminant | Exponentiation method | | | | |
|-----------------------------|---------------------------------|-----------------------|--------------------------|------------------------|------------------------------|----------------------------|
| | | EXP see [2] | CRIADexpPP see Prop.3 | CRIADexp see Prop.4 | CRIADexpwindow see Prop.5 | CRIADexpBGMW see Prop.6 |
| 20 | 100 | 0.50 | 0.38 | 0.33 | 0.44 | 0.11 |
| 40 | 100 | 1.10 | 0.94 | 0.88 | 0.93 | 0.28 |
| 60 | 100 | 2.14 | 1.82 | 1.64 | 1.71 | 0.39 |
| 80 | 100 | 3.52 | 2.85 | 2.75 | 2.68 | 0.60 |
| 100 | 100 | 5.71 | 4.18 | 4.12 | 3.79 | 0.82 |
| 20 | 200 | 0.99 | 0.77 | 0.60 | 0.66 | 0.17 |
| 40 | 200 | 2.09 | 1.70 | 1.37 | 1.49 | 0.27 |
| 60 | 200 | 3.68 | 2.91 | 2.47 | 2.47 | 0.55 |
| 80 | 200 | 5.66 | 4.23 | 3.90 | 3.68 | 0.72 |
| 100 | 200 | 8.68 | 5.99 | 5.66 | 5.21 | 1.10 |
| 20 | 300 | 1.76 | 0.87 | 0.83 | 0.99 | 0.27 |
| 40 | 300 | 3.35 | 1.92 | 1.93 | 1.81 | 0.44 |
| 60 | 300 | 5.33 | 3.35 | 3.18 | 2.91 | 0.82 |
| 80 | 300 | 8.13 | 5.00 | 4.83 | 4.23 | 1.04 |
| 100 | 300 | 11.53 | 7.91 | 6.76 | 6.37 | 1.48 |
| 20 | 400 | 1.76 | 1.48 | 1.10 | 1.32 | 0.32 |
| 40 | 400 | 3.79 | 3.07 | 2.36 | 2.59 | 0.55 |
| 60 | 400 | 6.64 | 4.89 | 3.96 | 4.06 | 0.88 |
| 80 | 400 | 10.10 | 7.25 | 5.99 | 5.88 | 1.20 |
| 100 | 400 | 14.88 | 10.00 | 8.73 | 8.35 | 1.70 |
| 20 | 500 | 3.35 | 2.58 | 1.38 | 1.43 | 0.43 |
| 40 | 500 | 7.14 | 4.61 | 2.97 | 2.91 | 0.66 |
| 60 | 500 | 10.88 | 7.30 | 5.06 | 4.61 | 1.15 |
| 80 | 500 | 15.98 | 10.27 | 7.75 | 6.96 | 1.65 |
| 100 | 500 | 22.52 | 13.84 | 10.44 | 9.66 | 2.09 |

References

1. I. Biehl and J. Buchmann: *Algorithms for Quadratic Orders*. Proceedings of Symposia in Applied Mathematics. **48**. American Mathematical Society: 1994. pp. 425-451.
2. I. Biehl, J. Buchmann and C. Thiel: *Cryptographic Protocols Based on Discrete Logarithms in Real-quadratic Orders*, Advances in Cryptology – CRYPTO '94, LNCS **839**, Springer, 1995, pp. 56 – 60
3. I. Biehl, B. Meyer and C. Thiel: *Cryptographic Protocols Based on Real-Quadratic A-fields*. Proceedings of ASIACRYPT '96. Springer: 1996. pp. 15-25.
4. E. Brickell, D. Gordon, K. McCurley, D. Wilson: *Fast Exponentiation with Pre-computation*, Advances in Cryptology, EUROCRYPT '92, LNCS **658**, Springer, 1993, pp. 200-207
5. J. Buchmann, M. Maurer: *Approximate Evaluation of $L(1, \chi_\Delta)$* , Technical Report, Darmstadt, University of Technology, 1997
6. J. Buchmann, C. Thiel, H.C. Williams: *Short representation of quadratic integers*, Computational Algebra and Number Theory, Mathematics and its Applications **325**, 1995, pp. 159 – 185
7. J. Buchmann and H.C. Williams: *A Key Exchange System Based on Real Quadratic Fields*. Proceedings of CRYPTO '89. Springer: 1989. pp. 335-343.

8. H. Cohen: *A Course in Computational Algebraic Number Theory*. Graduate Texts in Mathematics **138**. Springer: Berlin, 1993.
9. H. Cohen: *Analysis of the flexible window powering algorithm*, preprint available via <http://www.math.u-bordeaux.fr/~cohen/>
10. L.K. Hua: *Introduction to Number Theory*. Springer-Verlag: New York, 1982.
11. D. Hühnlein: *Quadratic orders for NESSIE - Overview and parameter sizes of three public key families*, submitted to ISSE 2000, preprint available via <http://www.informatik.tu-darmstadt.de/TI/Veroeffentlichung/TR/Welcome.html>
12. D. Hühnlein, M. Maurer, S. Paulus: *On the complexity and efficiency of cryptosystems using real quadratic number fields*, Technical report TU Darmstadt, to appear, 2000
13. M.J. Jacobson Jr.: *Subexponential Class Group Computation in Quadratic Orders*, PhD-thesis, TU Darmstadt, appeared in Shaker, Aachen, ISBN 3-8265-6374-3, 1999
14. D.E. Knuth: *The Art of Computer Programming*. Vol. 2: Seminumerical algorithms. Addison-Wesley, Reading MA, 1981.
15. H.W. Lenstra: *On the computation of regulators and class numbers of quadratic fields*, London Math. Soc. Lecture Notes, **56**, 1982, pp. 123-150
16. LiDIA: *A c++ library for algorithmic number theory*, via <http://www.informatik.tu-darmstadt.de/TI/LiDIA>
17. M. Maurer: *Regulator approximation and fundamental unit computation for real quadratic orders*, PhD-thesis, TU-Darmstadt, to appear 2000
18. R. Scheidler, J. Buchmann, H.C. Williams: *Implementation of a key exchange protocol using real quadratic fields (extended abstract)*, Advances in Cryptology – EU-ROCRYPT '90, Springer, LNCS **473**, 1991, pp. 98 – 109
19. R. Scheidler, J. Buchmann and H.C. Williams: *A Key-Exchange Protocol Using Real Quadratic Fields*. Journal of Cryptology **7**. 1994. pp. 171-199.
20. R.J. Schoof: *Quadratic Fields and Factorization*. In: H.W. Lenstra, R. Tijdeman, (eds.): *Computational Methods in Number Theory*. Math. Centrum Tracts **155**. Part II. Amsterdam, 1983. pp. 235-286.
21. D. Shanks, *The infrastructure of a real quadratic field and its applications*. Proc. Number Theory Conference, Boulder. 1972, pp. 217-224.
22. H.C. Williams: *A numerical investigation into the length of the period of the continued fraction expansion of \sqrt{D}* , Math. Comp. **36**, 1981, pp. 593-601

Root Finding Interpolation Attack

Kaoru Kurosawa, Tetsu Iwata, and Viet Duong Quang

Department of Communication and Integrated Systems,
Tokyo Institute of Technology,
2-12-1 O-okayama, Meguro-ku, Tokyo 152-8552, Japan
{kurosawa,tez,viet}@ss.titech.ac.jp

Abstract. In this paper, we first show that there are *several equivalent* keys for $t + 1$ chosen plaintexts if the degree of the reduced cipher is $t - 1$. This is against the claim by Jakobsen and Knudsen. We also derive an upper bound on the number of equivalent last round keys for $t + 1$ chosen plaintexts. We further show an efficient method which finds all the equivalent keys by using Rabin's root finding algorithm. We call our attack *root finding interpolation attack*.

Keywords: Block cipher, interpolation attack, root finding algorithm, resultant.

1 Introduction

Consider a Feistel type block cipher of block size $2n$ with a round function $F(K, x)$. For a fixed key K , $F(K, x)$ can be viewed as a polynomial $f_K(x)$ in x over $\text{GF}(2^n)$. The interpolation attack [4] succeeds if $\deg f_K(x)$ is small for any K and the number of rounds is not large. More precisely, suppose that the degree of the reduced cipher is $t - 1$, where the degree of the reduced cipher will be defined in Definition 2.1. Then

1. Jakobsen and Knudsen claimed that the last round key K_m can be recovered from $t + 1$ chosen plaintexts (see [4, Theorem 3]).
2. They used exhaustive search to find K_m .

On the other hand, given a polynomial $f(x)$ over $\text{GF}(p)$, Berlekamp proposed a probabilistic algorithm of finding a root $\alpha \in \text{GF}(p)$ of $f(x) = 0$ for any odd prime p [1]. Rabin generalized Berlekamp's algorithm to any finite fields [8]. In Rabin's algorithm, the expected number of bit operations to find a root of $f(x) = 0$ over $\text{GF}(2^n)$ is

$$O(n^2 d L(d) L(n)),$$

where $d = \deg f(x)$ and $L(n) = \log n \times \log \log n$.

In this paper, we first show that for $t + 1$ chosen plaintexts, there are *several equivalent* keys. This is against the claim by Jakobsen and Knudsen [4, Theorem 3]. We also derive an upper bound on the number of equivalent last round keys for $t + 1$ chosen plaintexts.

We next show an efficient method which finds all the equivalent last round keys K_m . We call our attack *root finding interpolation attack* because it uses Rabin's root finding algorithm [8]. By using more than $t + 1$ chosen plaintexts, we can uniquely determine K_m .

Further, Jakobsen and Knudsen claimed that the number of necessary chosen plaintexts can be smaller than $t + 1$ if they use the meet in the middle approach [4]. However, the number of equivalent keys increases if the number of chosen plaintexts decreases in general. Therefore, their claim cannot be justified. For this problem, we derive another upper bound on the number of equivalent last round keys for a certain number of chosen plaintexts which is less than $t + 1$.

Related works: Youssef and Gong studied the effect of the choice of the irreducible polynomial defining $\text{GF}(2^n)$ on $\deg f_K(x)$ and whether or not there exists a simple linear transformation on the input or output bits such that the resulting polynomial has a less degree [9].

The higher differential attack succeeds if the round function $F(K, x)$ can be expressed as a set of low degree Boolean functions [4]. Moriai et al. showed an improved higher differential attack for a 5 rounds CAST cipher in which K_m is computed by solving simultaneous linear equations [6].

2 Preliminaries

2.1 Notation

Consider an m round Feistel type block cipher with block size $2n$. Let $x = (x_L, x_R)$ denote the plaintext, where $x_L = (x_1, \dots, x_n)$ and $x_R = (x_{n+1}, \dots, x_{2n})$. Similarly, let $y = (y_L, y_R)$ denote the ciphertext. Let

$$C_0^L \triangleq x_L \text{ and } C_0^R \triangleq x_R .$$

The round function F operates as follows.

$$\begin{cases} C_i^L = C_{i-1}^R , \\ C_i^R = F(K_i, C_{i-1}^R) + C_{i-1}^L , \end{cases} \quad (1)$$

where K_i denotes the i -th round key. The ciphertext $y = (y_L, y_R)$ is given by (C_m^R, C_m^L) . See Fig. 1.

2.2 Reduced Cipher Assumption

We say that:

1. (C_{m-1}^R, C_{m-1}^L) is the reduced ciphertext and
2. (C_{m-2}^R, C_{m-2}^L) is the second reduced ciphertext, respectively.

Define

$$\begin{aligned} \tilde{y} &= (\tilde{y}_L, \tilde{y}_R) \triangleq (C_{m-1}^R, C_{m-1}^L) \\ \hat{y} &= (\hat{y}_L, \hat{y}_R) \triangleq (C_{m-2}^R, C_{m-2}^L). \end{aligned}$$

See Fig. 1.

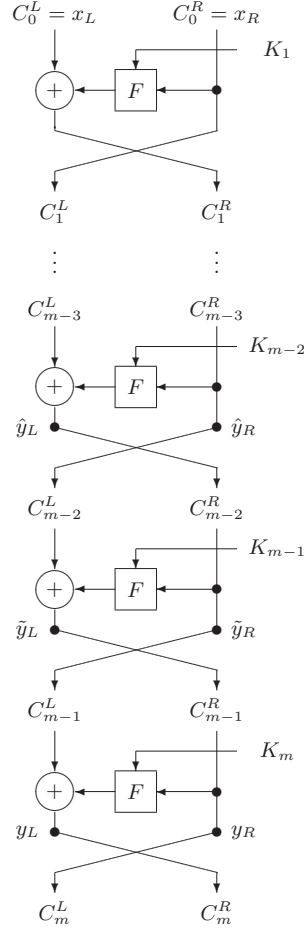


Fig. 1. The m round Feistel cipher

Definition 2.1. Fix the right half of a plaintext x_R as $x_R = 0$. Fix the key of the block cipher arbitrarily. Then we say that:

1. A block cipher satisfies the reduced cipher assumption of degree $t - 1$ if the right half \tilde{y}_R of the reduced ciphertext can be expressed as

$$\tilde{y}_R = f_1(x_L) \quad (2)$$

for some polynomial $f_1(x)$ over $GF(2^n)$ such that $\deg f_1(x) \leq t - 1$.

2. A block cipher satisfies the second reduced cipher assumption of degree $u - 1$ if the right half \hat{y}_R of the second reduced ciphertext can be expressed as

$$\hat{y}_R = f_2(x_L) \quad (3)$$

for some polynomial $f_2(x)$ over $GF(2^n)$ such that $\deg f_2(x) \leq u - 1$.

2.3 Lagrange Interpolation

Let Q be a field. Given $2t$ elements $x_1, \dots, x_t, y_1, \dots, y_t \in Q$, where the x_i s are distinct. Define

$$f(x) = \sum_{i=1}^t \lambda_i(x) y_i, \quad (4)$$

where

$$\lambda_i(x) = \prod_{1 \leq j \leq t, j \neq i} \frac{x - x_j}{x_i - x_j}.$$

Then $f(x)$ is the only polynomial over Q of degree at most $t - 1$ such that $f(x_i) = y_i$ for $i = 1, \dots, t$. Eq.(4) is known as the *Lagrange interpolation formula*.

3 Root Finding Algorithm over $\text{GF}(2^n)$

Given a polynomial $h(x)$ of degree d with coefficients in $\text{GF}(2^n)$, Rabin showed an efficient probabilistic polynomial time algorithm which computes a root of $h(x) = 0$ in $\text{GF}(2^n)$ if such a root does exist [8].

First compute

$$h_1(x) = \gcd(h(x), x^{2^n-1} - 1).$$

If $h_1(x) = 1$, then $h(x)$ has no roots in $\text{GF}(2^n)$. In general,

$$h_1(x) = (x - \alpha_1) \cdots (x - \alpha_k), \quad k \leq d,$$

where the α_i are the pairwise different roots in $\text{GF}(2^n)$ of $h(x) = 0$.

On the other hand, the trace function is defined as

$$\text{Tr}(x) = x + x^2 + \cdots + x^{2^{n-1}}.$$

For any $\alpha \in \text{GF}(2^n)$, it is known that

$$\text{Tr}(\alpha) = 0 \text{ or } 1.$$

Rabin first proved the following proposition

Proposition 3.1. *For any fixed $\alpha_1 \neq \alpha_2 \in \text{GF}(2^n)$, choose $r \in \text{GF}(2^n)$ randomly. Then*

$$\Pr(\text{Tr}(r\alpha_1) \neq \text{Tr}(r\alpha_2)) = \frac{1}{2}.$$

Rabin next showed the following root finding algorithm.

Let $h_0(x) = h_1(x)$.

Step 1. If $\deg h_0(x) = 1$, then we have found a root. Otherwise goto step 2.

Step 2. Choose $r \in \text{GF}(2^n)$ randomly. Compute

$$h_r(x) = \gcd(h_0(x), \text{Tr}(rx)).$$

Step 3. If $h_r(x) = 1$ or $h_0(x)$, goto step 2. Otherwise, let

$$h_0(x) := \begin{cases} h_r(x) & \text{if } \deg h_r(x) \leq \frac{1}{2} \deg h_0(x) \\ h_0(x)/h_r(x) & \text{otherwise.} \end{cases}$$

Goto Step 1.

From Proposition 3.1, it holds that

$$\Pr [0 < \deg h_r(x) < \deg h_1(x)] \geq \frac{1}{2}.$$

Therefore, it can be shown that [8] the expected number of bit operations is

$$O(n^2 d L(d) L(n)),$$

where

$$L(n) = \log n \times \log \log n.$$

4 Equivalent Keys and Root Finding Interpolation Attack

In this section, we first show that for $t + 1$ chosen plaintexts, there are *several equivalent* keys. This is against the claim by Jakobsen and Knudsen [4, Theorem 3]. We also derive an upper bound on the number of equivalent last round keys for $t + 1$ chosen plaintexts.

We next show an efficient method which finds all the equivalent keys. We call our attack *root finding interpolation attack* because it uses Rabin's root finding algorithm [8]. By using more than $t + 1$ chosen plaintexts, we can uniquely determine K_m .

For a plaintext $(x_L, x_R) = (x_i, 0)$, let $(y_{L,i}, y_{R,i})$ denote the ciphertext, $(\tilde{y}_{L,i}, \tilde{y}_{R,i})$ denote the reduced ciphertext and $(\hat{y}_{L,i}, \hat{y}_{R,i})$ denote the second reduced ciphertext.

4.1 Key Equation

In this subsection, we derive a key equation

$$h(K_m) = 0$$

in K_m such that $\deg h(K_m) \leq d$, where d is given below.

Definition 4.1. We say that the round function F satisfies K polynomial assumption of degree d if for any fixed x , there exists a polynomial g_x with $\deg g_x(K) \leq d$ such that

$$F(K, x) = g_x(K).$$

Suppose that there exists a block cipher which satisfies the *reduced cipher assumption of degree $t - 1$* . Further, without loss of generality, we can assume that there exists d such that the block cipher satisfies *K polynomial assumption of degree d* .

First by using the Lagrange formula, $f_1(x)$ of eq.(2) can be expressed as

$$f_1(x) = \lambda_1(x)f_1(x_1) + \cdots + \lambda_t(x)f_1(x_t)$$

for some polynomials $\lambda_1(x), \dots, \lambda_t(x)$, where each $\lambda_i(x)$ is determined by x_1, \dots, x_t . Then we have

$$f_1(x_{t+1}) = \lambda_1(x_{t+1})f_1(x_1) + \cdots + \lambda_t(x_{t+1})f_1(x_t)$$

for $x = x_{t+1}$. Substituting eq.(2) into the above equation yields that

$$\tilde{y}_{R,t+1} = \lambda_1(x_{t+1})\tilde{y}_{R,1} + \cdots + \lambda_t(x_{t+1})\tilde{y}_{R,t} \quad (5)$$

On the other hand, from eq.(1), it holds that

$$\tilde{y}_{R,i} = F(K_m, y_{R,i}) + y_{L,i}. \quad (6)$$

Substitute eq.(6) into eq.(5). Then we have

$$\begin{aligned} & F(K_m, y_{R,t+1}) + y_{L,t+1} \\ &= \lambda_1(x_{t+1})(F(K_m, y_{R,1}) + y_{L,1}) + \cdots + \lambda_t(x_{t+1})(F(K_m, y_{R,t}) + y_{L,t}) \end{aligned} \quad (7)$$

The above equation is rearranged as

$$h(K_m) = 0 \quad (8)$$

for some polynomial of K_m over $\text{GF}(2^n)$ such that

$$\deg h(K) \leq d$$

from *K polynomial assumption of degree d* . We call eq.(8) the key equation. (This equation is not redundant. That is, we cannot reduce $\deg h(K_m)$.)

4.2 Equivalent Keys

Jakobsen and Knudsen claimed that the last round key K_m can be recovered from $t+1$ chosen plaintexts in [4, Theorem 3]. However, eq.(8) implies that there are d or less equivalent keys. Now we have proved the following theorem.

Theorem 4.1. *Suppose that there exists a block cipher which satisfies the reduced cipher assumption of degree $t - 1$ and K polynomial assumption of degree d . Then for $t+1$ chosen plaintexts, there are d or less equivalent last round keys.*

In fact, the interpolation attack must require more than $t+1$ chosen plaintexts to uniquely determine K_m . If the round function $F(K, x)$ is not algebraically constructed, the situation is worse because d is usually large. In this case, there are many equivalent keys for $t+1$ chosen plaintexts and the interpolation attack will require many chosen plaintexts to uniquely determine K_m .

4.3 Root Finding Interpolation Attack

We propose an attack which efficiently finds all the equivalent keys K_m by solving eq.(8) by using Rabin's algorithm of Sec.3. By using more than $t + 1$ chosen plaintexts, we can uniquely determine K_m . We call this attack *root finding interpolation attack*.

First suppose that $t + 1$ chosen plaintext/ciphertext pairs are available such that the plaintexts are $(x_1, 0), \dots, (x_{t+1}, 0)$ and the ciphertexts are $(y_{L,1}, y_{R,1}), \dots, (y_{L,t+1}, y_{R,t+1})$. Then

Step 1. Compute the coefficients of $h(K_m)$ of eq.(8) from x_1, \dots, x_{t+1} and $(y_{L,1}, y_{R,1}), \dots, (y_{L,t+1}, y_{R,t+1})$. Especially, $\lambda_i(x)$ is determined by x_1, \dots, x_t though the Lagrange interpolation formula.

Step 2. Solve eq.(8) by using Rabin's algorithm of Sec.3. Then we obtain d or less equivalent keys K_m .

Next suppose that some extra (chosen plaintext, ciphertext) pairs are available. Then the set of equivalent keys is made smaller and we can finally uniquely determine K_m . An alternative way is as follows. Obtain two key equations $h_i(K_m) = 0$ for $i = 1, 2$ from $t + 2$ chosen plaintexts. Compute $\gcd(h_1(K_m), h_2(K_m))$. If K_m is uniquely determined from the gcd, then we have done. Otherwise, execute the same procedure for more chosen plaintexts.

5 On the Meet in the Middle Approach

Jakobsen and Knudsen also claimed that the number of necessary chosen plaintexts can be smaller than $t + 1$ if they use the meet in the middle approach [4]. However, the number of equivalent keys increases if the number of chosen plaintexts decreases in general. Therefore, their claim cannot be justified.

In this section, we derive an upper bound on the number of equivalent last round keys for certain number of chosen plaintexts which is less than $t + 1$.

Suppose that there exists a block cipher which satisfies the *second reduced cipher assumption of degree $u - 1$* and *K polynomial assumption of degree d* . Then from $u + 2$ chosen plaintexts, we first derive two equations on (K_{m-1}, K_m) such that

$$\begin{aligned} H_1(K_{m-1}, K_m) &= 0, \\ H_2(K_{m-1}, K_m) &= 0. \end{aligned}$$

We next compute the resultant

$$h(K_m) \triangleq R(H_1, H_2)$$

of H_1 and H_2 which yields that $\deg h(K_m) \leq 2d^3$. The above equation means that there are $2d^3$ or less equivalent last round keys for $u + 2$ chosen plaintexts.

Finally, we can find all the equivalent keys by solving $h(K_m) = 0$ by the Rabin's algorithm.

5.1 Resultant [10]

Let

$$A(x) = \sum_{i=0}^d a_i x^i$$

$$B(x) = \sum_{i=0}^e b_i x^i$$

be two polynomials over a field Q .

Define

$$R(A, B) = \det \left| \begin{array}{cccc} a_d & a_{d-1} & \cdots & a_0 \\ & a_d & a_{d-1} & \cdots & a_0 \\ & & \ddots & \ddots & \\ b_e & b_{e-1} & \cdots & b_0 \\ & b_e & b_{e-1} & \cdots & b_0 \\ & & \ddots & \ddots & \end{array} \right| \left. \begin{array}{l} \left. \begin{array}{c} \\ \\ \end{array} \right\} e \\ \left. \begin{array}{c} \\ \\ \end{array} \right\} d \end{array} \right.$$

We say that $R(A, B)$ is the resultant of $A(x)$ and $B(x)$.

Proposition 5.1. *$A(x)$ and $B(x)$ have a common root in Q if and only if*

$$R(A, B) = 0.$$

5.2 Key Equation

First by using the Lagrange formula, $f_2(x)$ of eq.(3) can be expressed as

$$f_2(x) = \delta_1(x)f_2(x_1) + \cdots + \delta_u(x)f_2(x_u)$$

for some polynomials $\delta_1(x), \dots, \delta_u(x)$, where each $\delta_i(x)$ is determined by x_1, \dots, x_u . Then we have

$$f_2(x_{u+1}) = \delta_1(x_{u+1})f_2(x_1) + \cdots + \delta_u(x_{u+1})f_2(x_u)$$

for $x = x_{u+1}$. Substituting eq.(3) into the above equation yields that

$$\hat{y}_{R,u+1} = \delta_1(x_{u+1})\hat{y}_{R,1} + \cdots + \delta_u(x_{u+1})\hat{y}_{R,u} \quad (9)$$

On the other hand, from eq.(1), it holds that

$$\begin{aligned} \hat{y}_{R,i} &= F(K_{m-1}, \tilde{y}_{R,i}) + \tilde{y}_{L,i} \\ &= F(K_{m-1}, F(K_m, y_{R,i}) + y_{L,i}) + y_{R,i}. \end{aligned} \quad (10)$$

Substitute eq.(10) into eq.(9). Then we have

$$H_1(K_{m-1}, K_m) = 0$$

such that

$$H_1(K_{m-1}, K_m) = \sum_{i=0}^d a_i(K_m) K_{m-1}^i$$

$$\deg a_i(K_m) \leq d^2$$

from K polynomial assumption.

Similarly for $x = x_{u+2}$, we have

$$H_2(K_{m-1}, K_m) = 0$$

such that

$$H_2(K_{m-1}, K_m) = \sum_{i=0}^d b_i(K_m) K_{m-1}^i$$

$$\deg b_i(K_m) \leq d^2$$

Finally, let

$$h(K_m) \triangleq R(H_1, H_2),$$

where $R(H_1, H_2)$ is the resultant of H_1 and H_2 . From Proposition 5.1, it holds that

$$h(K_m) = 0$$

since H_1 and H_2 have a common root. Further, we can see that

$$\deg h(K) \leq 2d^3$$

5.3 Equivalent Keys

From the previous subsection, we obtain the following theorem.

Theorem 5.1. *Suppose that there exists a block cipher which satisfies the second reduced cipher assumption of degree $u-1$ and K polynomial assumption of degree d . Then for $u+2$ chosen plaintexts, there are $2d^3$ or less equivalent last round keys.*

5.4 Root Finding Resultant Attack

We propose an attack such as follows which we call *root finding resultant attack*.

First suppose that $u+2$ chosen plaintext/ciphertext pairs are available such that the plaintexts are $(x_1, 0), \dots, (x_{u+2}, 0)$ and the ciphertexts are $(y_{L,1}, y_{R,1}), \dots, (y_{L,u+2}, y_{R,u+2})$. Then

Step 1. Compute the coefficients of H_1 and H_2 from x_1, \dots, x_{u+2} and $(y_{L,1}, y_{R,1}), \dots, (y_{L,u+2}, y_{R,u+2})$.

Step 2. Compute $h(K_m) = R(H_1, H_2)$.

Step 3. Solve $h(K_m) = 0$ by using Rabin's algorithm of Sec.3. Then we obtain $2d^3$ or less equivalent keys K_m .

Next suppose that some extra (chosen plaintext, ciphertext) pairs are available. Then the set of equivalent keys is made smaller and we can finally uniquely determine K_m . We can also have an alternative method similarly to Sec.4.3.

6 Example

The m round \mathcal{PURE} cipher [4] is defined by letting

$$F(K, x) = (K + x)^3$$

over $\text{GF}(2^{32})$.

Lemma 6.1. *In eq.(4),*

$$\lambda_1(x) + \cdots + \lambda_t(x) = 1.$$

Proof. $f(x) = 1$ is the only polynomial over Q of degree at most $t - 1$ such that $f(x_i) = 1$ for $i = 1, \dots, t$. Therefore, from eq.(4), we have

$$1 = \lambda_1(x) + \cdots + \lambda_t(x).$$

Q.E.D.

Corollary 6.1. *The m round \mathcal{PURE} cipher has two or less equivalent last round keys for $3^{m-3} + 2$ chosen plaintexts.*

Proof. Let $t - 1 = 3^{m-3}$ and $d = 3$. Then Theorem 4.1 tells us that there are $d = 3$ or less equivalent last round keys for $t + 1 = 3^{m-3} + 2$ chosen plaintexts. However, in this case, eq.(7) is written as follows.

$$\begin{aligned} & (K_m + y_{R,t+1})^3 + y_{L,t+1} \\ &= \lambda_1(x_{t+1})((K_m + y_{R,1})^3 + y_{L,1}) + \cdots + \lambda_t(x_{t+1})((K_m + y_{R,t})^3 + y_{L,t}). \end{aligned}$$

By rearranging the above equation, we obtain the key equation $h(K_m)$ such that

$$\deg h(K_m) = 2$$

because the coefficient of K_m^3 is canceled from lemma 6.1. This implies that there are two or less equivalent last round keys.

Q.E.D.

The proposed attack computes all the equivalent keys K_m by solving the quadratic equation $h(K_m) = 0$ over $\text{GF}(2^{32})$. By using one more chosen plaintext, we can uniquely determine K_m .

On the contrary, Jakobsen and Knudsen claimed that the interpolation attack needs $3^{m-3} + 2$ chosen plaintexts to recover K_m .

Corollary 6.2. *The m round \mathcal{PURE} cipher has 54 or less equivalent last round keys for $3^{m-4} + 3$ chosen plaintexts.*

Proof. Let $u - 1 = 3^{m-4}$ and $d = 3$. Then Theorem 5.1 tells us that there are $2d^3 = 54$ or less equivalent last round keys for $u + 2 = 3^{m-4} + 3$ chosen plaintexts. Q.E.D.

7 Summary

In this paper, we first showed that for $t + 1$ chosen plaintexts, there are several equivalent last round keys if the degree of the reduced cipher is $t - 1$. This is against the claim by Jakobsen and Knudsen on interpolation attack [4, Theorem 3]. We also derived an upper bound on the number of equivalent last round keys for $t + 1$ chosen plaintexts.

We next showed an efficient method which finds all the equivalent last round keys K_m . We call our attack *root finding interpolation attack* because it uses Rabin's root finding algorithm [8]. By using more than $t + 1$ chosen plaintexts, we can uniquely determine K_m .

The number of equivalent keys increases if the number of chosen plaintexts decreases in general. For this problem, we derived another upper bound on the number of equivalent last round keys for a certain number of chosen plaintexts which is less than $t + 1$.

As an example, we showed that the m round \mathcal{PURE} cipher has two or less equivalent last round keys for $3^{m-3} + 2$ chosen plaintexts and 54 or less equivalent last round keys for $3^{m-4} + 3$ chosen plaintexts. The proposed attack efficiently computes all the equivalent keys K_m by solving a key equation $h(K_m) = 0$ over $\text{GF}(2^{32})$ by using Rabin's root finding algorithm. By using more chosen plaintext, we can uniquely determine K_m .

It will be interesting if we can extend our method to the probabilistic interpolation attack [3] which succeeds even if $F(K, x)$ is approximated by a low degree polynomial.

References

1. E.R.Berlekamp. Factoring polynomials over large finite fields. In *Math. Comput.*, vol. 24, pp. 713–735, 1970.
2. E.Biham and A.Shamir. Differential Cryptanalysis of the Data Encryption Standard. Springer-Verlag, 1993.
3. T.Jakobsen. Cryptanalysis of block ciphers with probabilistic non-linear relations of low degree. In *Advances in Cryptology — CRYPTO' 98 Proceedings*, volume 1462 of *Lecture Notes in Computer Science*, pages 212–222, Springer-Verlag, 1998.
4. T.Jakobsen and L.R.Knudsen. The interpolation attack on block ciphers. In *Fast Software Encryption*, volume 1267 of *Lecture Notes in Computer Science*, pages 28–40, Springer-Verlag, January 1997.
5. M.Matsui. Linear cryptanalysis method for DES cipher. In *Advances in Cryptology — EUROCRYPT' 93 Proceedings*, volume 765 of *Lecture Notes in Computer Science*, pages 386–397, Springer-Verlag, 1993.

6. S.Moriai, T.Shimoyama and T.Kaneko. Higher order differential attack of a CAST cipher. Proc. of FSE '98, LNCS 1372, pp.17–32, (1998)
7. K.Nyberg and L.R.Knudsen. Provable security against a differential attack. In *Journal of Cryptology*, volume 8, number 1, pages 27–37, Winter 1995.
8. M.Rabin. Probabilistic algorithms in finite fields. SIAM Journal on Computing, vol.9, no.2, pp.273–280 (1980)
9. A.M.Youssef and G.Gong. On the interpolation attacks on block ciphers. Preproc. of FSE 2000, (2000)
10. R.Zippel. Effective polynomial computation. Kluwer Academic Publishers (1993)

Practical Security Evaluation against Differential and Linear Cryptanalyses for Feistel Ciphers with SPN Round Function

Masayuki Kanda

NTT Information Sharing Platform Laboratories,
1-1-612A Hikarinooka, Yokosuka-shi, Kanagawa, 239-0847, Japan
kanda@isl.ntt.co.jp

Abstract. This paper studies the upper bounds of the maximum differential and linear characteristic probabilities of Feistel ciphers with SPN round function. In the same way as for SPN ciphers, we consider the minimum number of differential and linear active s -boxes, which provides a measure of the upper bounds of these probabilities, in order to evaluate the security against differential and linear cryptanalyses. The purpose of this work is to clarify the (lower bound of) minimum numbers of active s -boxes in some consecutive rounds of Feistel ciphers, i.e., in three, four, six, eight, and twelve consecutive rounds, using differential and linear branch numbers \mathcal{P}_d , \mathcal{P}_l , respectively. Furthermore, we investigate the necessary condition for desirable P -functions, which means that the round functions are invulnerable to both differential and linear cryptanalyses. As an example, we show the round function of Camellia, which satisfies the condition.

1 Introduction and Motivation

The best known attacks are differential cryptanalysis [6] proposed by Biham and Shamir and linear cryptanalysis [13] proposed by Matsui. Since these cryptanalyses are the most powerful approaches known for attacking many symmetric block ciphers, designers should evaluate the security of any new proposed ciphers against differential and linear cryptanalyses. To do this it is necessary to determine the maximum differential and linear probabilities by a useful (and acceptable) method. Feistel ciphers are commonly analyzed by (a) the upper bounds of the maximum average of differential and linear hull probabilities or (b) the maximum differential and linear characteristic probabilities. SPN ciphers, on the other hand, are commonly analyzed by (c) the upper bounds of the maximum differential and linear characteristic probabilities. Recently, Hong et al. showed (a) the upper bounds of the maximum average of differential and linear hull probabilities of SPN ciphers [9].

With reference to method (a), Nyberg and Knudsen showed that the maximum average of differential and linear hull probabilities for r -round ($r \geq 4$) Feistel ciphers are bounded by $2p^2$, $2q^2$ if the maximum differential and linear

probabilities of the round function are p , q , respectively¹ [18]. They stated that Feistel ciphers are *provably secure* against differential and linear cryptanalyses if these probabilities are sufficiently low. This means that they are theoretically invulnerable to differential and linear cryptanalyses, since these probabilities are the upper bounds of the average of differential and linear hull probabilities. However, this approach has one fatal disadvantage. That is, these probabilities settle at some constant value even if the number of rounds increases. Therefore, a round function has to yield extremely low maximum differential and linear probabilities. This imposes a hard restriction on designing the round function. As a matter of fact, for a commercial cipher, MISTY [15] is provably secure with respect to differential and linear cryptanalyses.

Method (b) has been used to estimate many (extended) Feistel ciphers such as DES [6,13] and FEAL [16,2]. Biham and Shamir claimed that the higher the differential characteristic probability is, the higher the success rate of differential cryptanalysis is. This is because they exploited a single path between plaintexts and ciphertexts which holds significant differential characteristic probability. Matsui also claimed the same for linear cryptanalysis. Thus, Feistel ciphers are *sufficiently secure* against differential and linear cryptanalyses if these probabilities are less than the security threshold. Strictly speaking, however, these probabilities only give the lower bounds of the maximum average of differential and linear hull probabilities, since this method does not consider multiple paths between the same plaintexts and ciphertexts [12,17].

For SPN ciphers, Rijmen et al. introduced the branch number \mathcal{B} [19]. The number \mathcal{B} is the minimum number of active s -boxes in two consecutive rounds of a non-trivial differential characteristic or a non-trivial linear trail. Since each active s -box reduces the differential and linear characteristic probabilities, the number \mathcal{B} provides the upper bounds of the maximum differential and linear characteristic probabilities in two consecutive rounds. The security against differential and linear cryptanalyses is evaluated by piling up the number \mathcal{B} every two rounds. It is noted that Knudsen proposed a very similar concept for Feistel ciphers [10]. He noted that Feistel ciphers are *practically secure* against differential and linear cryptanalyses if the upper bounds of the maximum differential and linear characteristic probabilities are less than the security threshold.

It is obvious that the upper bounds of the maximum differential and linear characteristic probabilities by method (c) lie between the upper bounds of the maximum average of differential and linear hull probabilities by method (a) and the maximum differential and linear characteristic probabilities by method (b). Moreover, for most ciphers, the maximum averages of differential and linear hull probabilities, which provide the actual invulnerability to differential and linear cryptanalyses, are much lower than the upper bounds of these probabilities if the number of rounds increases. Therefore, it is worth investigating the upper bounds of the maximum differential and linear characteristic probabilities.

¹ Aoki and Ohta showed that these probabilities are bounded by p^2 , q^2 if the round function is bijective and $r \geq 3$ [3]

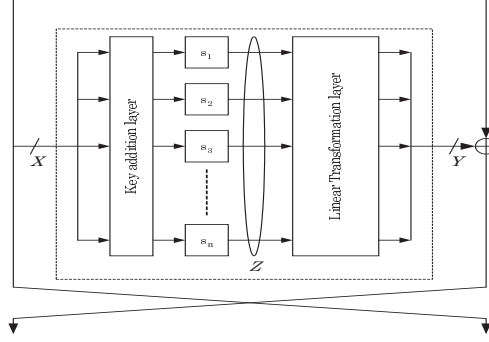


Fig. 1. SPN round function

since we assume that the round key, which is used within one round, consists of independent and uniformly random bits, and is bitwise XORed with data.

Notations describe the model as below.

S -function is a non-linear transformation layer with n parallel m -bit bijective s -boxes. That is,

$$\begin{aligned} S : (\mathbb{Z}_2^m)^n &\longrightarrow (\mathbb{Z}_2^m)^n \\ X = (x_1, \dots, x_n) &\longmapsto Z = S(X) = (s_1(x_1), \dots, s_n(x_n)) \end{aligned}$$

P -function is a linear transformation layer, i.e.,

$$\begin{aligned} P : (\mathbb{Z}_2^m)^n &\longrightarrow (\mathbb{Z}_2^m)^n \\ Z = (z_1, \dots, z_n) &\longmapsto Y = P(Z) = (y_1, \dots, y_n) \end{aligned}$$

Finally, the SPN round function can be described as follows.

$$\begin{aligned} F : (\mathbb{Z}_2^m)^n &\longrightarrow (\mathbb{Z}_2^m)^n \\ X = (x_1, \dots, x_n) &\longmapsto Y = F(X) = P(S(X)) = (y_1, \dots, y_n) \end{aligned}$$

Let $X^{(i)}$ be the input data to the i -th round function, and $Y^{(i)}$ be the i -round output data. The Feistel cipher is defined as:

$$X^{(i+1)} = X^{(i-1)} \oplus Y^{(i)} \quad (1 \leq i \leq r),$$

where $(X^{(1)}|X^{(0)})$ is a plaintext and $(X^{(r)}|X^{(r+1)})$ is a ciphertext.

2.3 Definitions

We use the following definitions in this paper.

Definition 1. For any given $\Delta x, \Delta z, \Gamma x, \Gamma z \in \mathbb{Z}_2^m$, the differential and linear probabilities of each s -box s_i are defined as:

$$\begin{aligned} DP^{s_i}(\Delta x \rightarrow \Delta z) &= \frac{\#\{x \in \mathbb{Z}_2^m | s_i(x) \oplus s_i(x \oplus \Delta x) = \Delta z\}}{2^m} \\ LP^{s_i}(\Gamma z \rightarrow \Gamma x) &= \left(2 \times \frac{\#\{x \in \mathbb{Z}_2^m | x \cdot \Gamma x = s_i(x) \cdot \Gamma z\}}{2^m} - 1 \right)^2 \end{aligned}$$

Definition 2. The maximum differential and linear probabilities of s -boxes are defined as:

$$p_s = \max_i \max_{\Delta x \neq 0, \Delta z} DP^{s_i}(\Delta x \rightarrow \Delta z)$$

$$q_s = \max_i \max_{\Gamma x, \Gamma z \neq 0} LP^{s_i}(\Gamma z \rightarrow \Gamma x)$$

This means that p_s, q_s are the upper bounds of the maximum differential and linear probabilities for all s -boxes.

Definition 3. A differential active s -box is defined as an s -box given a non-zero input difference, while a linear active s -box is defined as an s -box given a non-zero output mask value [11].

Note: When an s -box is bijective, s -boxes given a non-zero output difference and a non-zero input mask value are also differential and linear active s -boxes, respectively.

Definition 4. Let $X = (x_1, \dots, x_n) \in \text{GF}(2^m)^n$ then the Hamming weight of X is denoted by

$$H_w(X) = \#\{i | x_i \neq 0\}.$$

This means that the Hamming weight of X equals the number of non-zero m -bit characters from $\text{GF}(2^m)$ of X .

3 Previous Works – the Security of SPN Ciphers

As mentioned above, the security of most SPN ciphers against differential and linear cryptanalyses is evaluated using the (lower bound of) minimum number of differential and linear active s -boxes, which are a measure of the upper bounds of differential and linear characteristic probabilities [19,8,5]. To determine the (lower bound of) minimum number of active s -boxes, Rijmen et al. defined the branch number \mathcal{B} [19].

Definition 5. In SPN ciphers, the differential branch number \mathcal{B}_d is defined as:

$$\mathcal{B}_d = \min_{\Delta X \neq 0} (H_w(\Delta X) + H_w(\theta(\Delta X))),$$

where ΔX is an input difference into the diffusion layer and $\theta(\Delta X)$ is an output difference from the layer.

Note that ΔX is also an output difference from a substitution layer and $\theta(\Delta X)$ is also an input difference to the next substitution layer. Since s -boxes are bijective, $H_w(\Delta X)$ equals the number of differential active s -boxes in the substitution layer and $H_w(\theta(\Delta X))$ equals that in the next substitution layer. That is, if n_d is the minimum number of differential active s -boxes in two consecutive rounds, then $n_d = \mathcal{B}_d$. Thus, it turns out that the minimum number of differential active s -boxes in $2r$ -round SPN ciphers is lower bounded by $r\mathcal{B}_d$, and the following theorem is obtained.

Theorem 1. *The maximum differential characteristic probability for $2r$ -round SPN cipher, $p_d^{(2r)}$, is upper bounded by $p_s^{(rB_d)}$.*

From the duality between differential characteristics and linear approximations [4,14], the following definition and theorem also are established.

Definition 6. *The linear branch number B_l is defined as:*

$$B_l = \min_{\Gamma Y \neq 0} (H_w(\theta^*(\Gamma Y)) + H_w(\Gamma Y)),$$

where ΓY is an output mask value of the diffusion layer θ and $\theta^*(\Gamma Y)$ is an input mask value of the layer. θ^* is the diffusion function of mask values concerning the layer.

Theorem 2. *The maximum linear characteristic probability for $2r$ -round SPN cipher, $q_l^{(2r)}$, is upper bounded by $q_s^{(rB_l)}$.*

4 Upper Bound of Differential Characteristic Probability

In this section, we investigate the upper bound of differential characteristic probability of Feistel cipher with SPN round function. In the same way as in the previous section, our goal is to clarify the (lower bound of) minimum number of differential active s -boxes in some consecutive rounds of Feistel cipher.

First, we show the useful lemma concerning the hamming weight for Feistel ciphers.

Lemma 1. *In Feistel ciphers, the following relationship holds.*

$$H_w(\Delta Y^{(i)}) = H_w(\Delta X^{(i-1)} \oplus \Delta X^{(i+1)}) \leq H_w(\Delta X^{(i-1)}) + H_w(\Delta X^{(i+1)})$$

Proof.

$$\begin{aligned} H_w(\Delta Y^{(i)}) &= H_w(\Delta X^{(i-1)} \oplus \Delta X^{(i+1)}) \\ &= \#\{s | \Delta x_s^{(i-1)} \neq 0 \text{ and } \Delta x_s^{(i+1)} = 0\} \\ &\quad + \#\{t | \Delta x_t^{(i-1)} = 0 \text{ and } \Delta x_t^{(i+1)} \neq 0\} \\ &\quad + \#\{u | \Delta x_u^{(i-1)} \neq 0 \text{ and } \Delta x_u^{(i+1)} \neq 0 \text{ and } x_u^{(i-1)} \neq x_u^{(i+1)}\} \\ &\leq H_w(\Delta X^{(i-1)}) + \#\{t | \Delta x_t^{(i-1)} = 0 \text{ and } \Delta x_t^{(i+1)} \neq 0\} \\ &\leq H_w(\Delta X^{(i-1)}) + H_w(\Delta X^{(i+1)}) \end{aligned}$$

Q.E.D.

Since there is a linear transformation layer (P -function) in the SPN round function, we will define the differential branch number \mathcal{P}_d in the same way as in the previous section. Note that it is obvious that if S -function is bijective then $H_w(\Delta X) = H_w(\Delta Z)$, since Δz_i also becomes a non-zero output difference through the differential active s_i -box.

Definition 7. If S -function is bijective, the differential branch number \mathcal{P}_d is defined as follows.

$$\mathcal{P}_d = \min_{\Delta X \neq 0} (H_w(\Delta X) + H_w(\Delta Y))$$

Here, we will define the upper bound of the maximum differential characteristic probability of Feistel cipher with SPN round function in the same way as used for the SPN cipher. That is, the upper bound of the probability is shown by the (lower bound of) minimum number of differential active s -boxes.

Definition 8. Assume Feistel cipher with SPN round function. Let $H_w(\Delta X^{(i)})$ be the number of the i th-round differential active s -boxes, then the differential characteristic probability of the r -round Feistel cipher, $p_d^{(r)}$, satisfies the following relationship.

$$p_d^{(r)} \leq p_s^{\min_{(\Delta X^{(0)}, \Delta X^{(1)}, \dots, \Delta X^{(r+1)}) \neq (0, 0, \dots)}} \sum_{i=1}^r H_w(\Delta X^{(i)})$$

From this definition, clarifying the upper bound of the maximum differential characteristic probability becomes equivalent to showing the (lower bound of) minimum number of differential active s -boxes. To discuss the minimum number easily after this, it is denoted as follows.

$$\mathcal{D}^{(r)} = \min_{(\Delta X^{(0)}, \Delta X^{(1)}, \dots, \Delta X^{(r+1)}) \neq (0, 0, \dots)} \sum_{i=1}^r H_w(\Delta X^{(i)})$$

Hereafter, because of limitations of space, we assume P -function is bijective. Note that this leads to $\mathcal{P}_d \geq 2$.

Lemma 2. The minimum number of differential active s -boxes in any three consecutive rounds satisfies $\mathcal{D}^{(3)} \geq 2$.

Proof. If $\Delta X^{(i)} = 0$, then $\Delta Y^{(i)} = 0$ and $\Delta X^{(i-1)} = \Delta X^{(i+1)} \neq 0$. This leads to $\mathcal{D}_1^{(3)} = 2 \times H_w(\Delta X^{(i-1)}) \geq 2$. On the other hand, If $\Delta X^{(i)} \neq 0$, it follows that $\mathcal{D}_2^{(3)} \geq H_w(\Delta X^{(i)}) + H_w(\Delta Y^{(i)}) \geq \mathcal{P}_d$, since Lemma 1 shows $H_w(\Delta X^{(i-1)}) + H_w(\Delta X^{(i+1)}) \geq H_w(\Delta Y^{(i)})$.

Q.E.D.

Lemma 3. The minimum number of differential active s -boxes in any four consecutive rounds satisfies $\mathcal{D}^{(4)} \geq \mathcal{P}_d$.

Proof. Without loss of generality, we assume that the four consecutive rounds run from the first round to the fourth round.

At no time do both input differences into any consecutive two rounds equal zero. In addition, by the assumption, at no time also do both input differences of every two rounds equal zero. Thus we only consider the six following cases concerning input differences into the consecutive four rounds.

$$(1) \Delta X^{(1)} \neq 0, \Delta X^{(2)} \neq 0, \Delta X^{(3)} \neq 0, \Delta X^{(4)} \neq 0$$

- (2) $\Delta X^{(1)} = 0, \Delta X^{(2)} \neq 0, \Delta X^{(3)} \neq 0, \Delta X^{(4)} \neq 0$
- (3) $\Delta X^{(1)} \neq 0, \Delta X^{(2)} = 0, \Delta X^{(3)} \neq 0, \Delta X^{(4)} \neq 0$
- (4) $\Delta X^{(1)} \neq 0, \Delta X^{(2)} \neq 0, \Delta X^{(3)} = 0, \Delta X^{(4)} \neq 0$
- (5) $\Delta X^{(1)} \neq 0, \Delta X^{(2)} \neq 0, \Delta X^{(3)} \neq 0, \Delta X^{(4)} = 0$
- (6) $\Delta X^{(1)} = 0, \Delta X^{(2)} \neq 0, \Delta X^{(3)} \neq 0, \Delta X^{(4)} = 0$

In case (1), by Lemma 2, $\mathcal{D}_1^{(4)} = \mathcal{D}_2^{(3)} + H_w(\Delta X^{(4)}) \geq \mathcal{P}_d + H_w(\Delta X^{(4)}) \geq \mathcal{P}_d + 1$.

In case (2), $\Delta X^{(1)} = 0$ leads to $\Delta Y^{(2)} = \Delta X^{(3)}$. Thus, $\mathcal{D}_2^{(4)} = H_w(\Delta X^{(2)}) + H_w(\Delta Y^{(2)}) + H_w(\Delta X^{(4)}) \geq \mathcal{P}_d + H_w(\Delta X^{(4)}) \geq \mathcal{P}_d + 1$.

Similarly, in cases (3), (4), and (5), we get $\mathcal{D}^{(4)} \geq \mathcal{P}_d + 1$.

In case (6), by Lemma 2, $\mathcal{D}_6^{(4)} = \mathcal{D}_2^{(3)} \geq \mathcal{P}_d$.

Q.E.D.

From the above proof, the following corollary is obtained.

Corollary 1. *The minimum number of differential active s-boxes in any four consecutive rounds satisfies*

(i) $\mathcal{D}^{(4)} \geq \mathcal{P}_d$, if and only if the input differences in both the first round and the fourth round are zero.

(ii) $\mathcal{D}^{(4)} \geq \mathcal{P}_d + 1$ in the other cases.

Lemma 4. *The minimum number of differential active s-boxes in any six consecutive rounds satisfies $\mathcal{D}^{(6)} \geq \mathcal{P}_d + 2$.*

Proof. – If $\Delta X^{(2)} \neq 0$ and $\Delta X^{(5)} \neq 0$, by Lemma 2, $\mathcal{D}_1^{(6)} = \mathcal{D}_2^{(3)} + \mathcal{D}_2^{(3)} \geq 2 \times \mathcal{P}_d$.

– If $\Delta X^{(2)} = \Delta X^{(5)} = 0$, we get $\Delta X^{(1)} = \Delta X^{(3)}$ and $\Delta Y^{(3)} = \Delta X^{(4)} = \Delta X^{(6)}$. Thus, $\mathcal{D}_2^{(6)} = 2 \times (H_w(\Delta X^{(3)}) + H_w(\Delta X^{(4)})) = 2 \times (H_w(\Delta X^{(3)}) + H_w(\Delta Y^{(3)})) \geq 2 \times \mathcal{P}_d$

– If $\Delta X^{(2)} = 0$ and $\Delta X^{(5)} \neq 0$, or $\Delta X^{(2)} \neq 0$ and $\Delta X^{(5)} = 0$, then $\mathcal{D}_3^{(6)} = \mathcal{D}_1^{(3)} + \mathcal{D}_2^{(3)} \geq \mathcal{P}_d + 2$ by Lemma 2.

Q.E.D.

Lemma 5. *The minimum number of differential active s-boxes in any eight consecutive rounds satisfies $\mathcal{D}^{(8)} \geq 2 \times \mathcal{P}_d + 1$.*

Proof. Again, corollary 1 shows that, in any four consecutive rounds, the minimum number of differential active s-boxes satisfies (i) $\mathcal{D}^{(4)} \geq \mathcal{P}_d$, if and only if the input differences in both the first round and the fourth round are zero, and $\mathcal{D}^{(4)} \geq \mathcal{P}_d + 1$ in the other cases.

Since there is no case in which both input differences into any two consecutive rounds are zero at the same time, the input differences in both the fourth and fifth rounds cannot be zero. That is, the eight consecutive rounds cannot be divided into two cases (i). Thus, $\mathcal{D}^{(8)} \geq \mathcal{P}_d + (\mathcal{P}_d + 1) \geq 2 \times \mathcal{P}_d + 1$.

Q.E.D.

Lemma 6. *The minimum number of differential active s -boxes in any twelve consecutive rounds satisfies $\mathcal{D}^{(12)} \geq 3 \times \mathcal{P}_d + 1$.*

Proof. $\mathcal{D}^{(12)}$ can be converted to three expressions, i.e., $4 \times \mathcal{D}^{(3)}$, $2 \times \mathcal{D}^{(6)}$, and $\mathcal{D}^{(8)} + \mathcal{D}^{(4)}$. Since $\mathcal{D}^{(12)}$ satisfies the three evaluations at the same time, $\mathcal{D}^{(12)} = \max\{4 \times \mathcal{D}^{(3)}, 2 \times \mathcal{D}^{(6)}, \mathcal{D}^{(8)} + \mathcal{D}^{(4)}\} \geq \mathcal{D}^{(8)} + \mathcal{D}^{(4)} \geq 3 \times \mathcal{P}_d + 1$.

Q.E.D.

From the proofs of above-mentioned lemmas, the useful theorem for the $4r$ -round Feistel ciphers is established as follows.

Theorem 3. *The minimum number of differential active s -boxes $\mathcal{D}^{(4r)}$ for $4r$ -round Feistel ciphers with SPN round function satisfies $\mathcal{D}^{(4r)} \geq r \times \mathcal{P}_d + \lfloor r/2 \rfloor$.*

Knudsen argued that for a Feistel cipher to be practically secure against differential and linear cryptanalyses, the upper bounds of the maximum differential and linear characteristic probabilities must be less than the security threshold. Generally speaking, the security threshold is equated to the inverse of the number of all plaintext blocks, i.e., 2^{-64} for 64-bit ciphers and 2^{-128} for 128-bit ciphers.

For example, let the maximum differential probability of an 8-bit s -box be $p_s = 2^{-6}$ and the differential branch number be $\mathcal{P}_d = 5$. It follows that 18-round Feistel ciphers, such as Camellia [1], are practically secure against differential cryptanalysis because of the following corollary.

Corollary 2. *Assuming that the round function consists of s -boxes yielding the maximum differential probability $p_s = 2^{-6}$ and P -function yielding the differential branch number $\mathcal{P}_d = 5$, then a 128-bit Feistel cipher with more than 16-rounds has no effective differential characteristic.*

Proof. By Definition 8 and Theorem 3, $p_d^{(16)} \leq (2^{-6})^{4 \times 5 + 2} = 2^{-132} < 2^{-128}$.

Q.E.D.

5 Upper Bound of Linear Characteristic Probability

In this section, the upper bound of linear characteristic probability is derived in the same way as in the previous section. That is, our goal is to clarify the (lower bound of) minimum number of linear active s -boxes in some consecutive rounds of Feistel cipher using the duality of differential characteristic and linear approximation.

First, the following theorem is established.

Theorem 4. *Consider a Feistel cipher with SPN round function. If the linear transformation layer P (P -function) is bijective, the cipher can be transformed into a Feistel cipher with the PSN round function.*

Proof. From the assumption that P -function is bijective, let describe $P(Z)$ as the transformation of Z by the P -function, and $P^{-1}(Z)$ as that by the inverse function of the P -function.

As mentioned above, in a Feistel cipher with SPN round function, the equation, $X^{(i+1)} = X^{(i-1)} \oplus P(S(X^{(i)}))$, is satisfied. Now, let $V^{(i)} = P^{-1}(X^{(i)})$. The above equation can be transformed as follows, since $C = A \oplus P(B) \Leftrightarrow C = P(P^{-1}(A) \oplus B)$ for any (A, B, C) .

$$\begin{aligned} X^{(i+1)} = X^{(i-1)} \oplus P(S(X^{(i)})) &\Leftrightarrow X^{(i+1)} = P(P^{-1}(X^{(i-1)}) \oplus S(X^{(i)})) \\ &\Leftrightarrow P(V^{(i+1)}) = P(V^{(i-1)} \oplus S(P(V^{(i)}))) \\ &\Leftrightarrow V^{(i+1)} = V^{(i-1)} \oplus S(P(V^{(i)})) \end{aligned}$$

The equation, $V^{(i+1)} = V^{(i-1)} \oplus S(P(V^{(i)}))$, denotes a Feistel cipher with the PSN round function. Accordingly, the ciphertext $(X^{(r)}, X^{(r+1)})$ obtained by applying a Feistel cipher with SPN round function to a plaintext $(X^{(1)}, X^{(0)})$ is equivalent to the result of changing the plaintext $(X^{(1)}, X^{(0)})$ to $(V^{(1)}, V^{(0)})$ by the P^{-1} -function first, then getting $(V^{(r)}, V^{(r+1)})$ from $(V^{(1)}, V^{(0)})$ from the Feistel cipher with PSN round function, and finally transforming it into the ciphertext $(X^{(r)}, X^{(r+1)})$ by the P -function.

Q.E.D.

Starting with the duality between differential characteristic and linear approximation, we will define the linear branch number \mathcal{P}_l , which is similar to the differential branch number \mathcal{P}_d . Hereafter, we assume P -function is bijective.

Definition 9. The linear branch number \mathcal{P}_l is defined as:

$$\mathcal{P}_l = \min_{\Gamma Y \neq 0} (H_w(P^*(\Gamma Y)) + H_w(\Gamma Y)) = \min_{\Gamma Y \neq 0} (H_w(\Gamma Z) + H_w(\Gamma Y)),$$

where ΓY , ΓZ is an output mask value and an input mask value of the P -function, respectively, and P^* is a diffusion function of mask values concerning the P -function.

Next, we will define the upper bound of the linear characteristic probability of a Feistel cipher with SPN round function. That is, the upper bound of the probability is shown by the (lower bound of) minimum number of linear active s -boxes.

Definition 10. Assume a Feistel cipher with SPN round function. If $H_w(\Gamma Z^{(i)})$ is the number of the i th-round linear active s -boxes, then the linear characteristic probability of the r -round Feistel cipher satisfies the following relationship.

$$p_l^{(r)} \leq p_s^{\min_{(\Gamma Y^{(0)}, \dots, \Gamma Y^{(r)}, \Gamma Y^{(r+1)}) \neq (\dots, 0, 0)} \sum_{i=1}^r H_w(\Gamma Z^{(i)})},$$

where $\Gamma Z^{(i)} = P^*(\Gamma Y^{(i)})$ and P^* is the diffusion function of mask values concerning the P -function.

From this definition, clarifying the upper bound of the linear characteristic probability becomes equivalent to determining the (lower bound of) minimum number of linear active s -boxes. To discuss the minimum number easily after this, we denote it as follows.

$$\mathcal{L}^{(r)} = \min_{(\Gamma Y^{(0)}, \dots, \Gamma Y^{(r)}, \Gamma Y^{(r+1)}) \neq (\dots, 0, 0)} \sum_{i=1}^r H_w(\Gamma Z^{(i)})$$

Theorem 5. Assume a Feistel cipher with SPN round function. If both S -function and P -function are bijective, then $\mathcal{L}^{(r)}$ and \mathcal{P}_l also satisfy Lemma 2 to Lemma 6 and Theorem 3.

Proof. Because of the bijective P -function, a Feistel cipher with SPN round function is transformed into one with PSN round function by Theorem 4. The cipher can be described as:

$$V^{(i+1)} = V^{(i-1)} \oplus S(P(V^{(i)})) = V^{(i-1)} \oplus S(X^{(i)}) = V^{(i-1)} \oplus Z^{(i)},$$

where $V^{(i)} = P^{-1}(X^{(i)})$, $Z^{(i)} = S(X^{(i)})$.

From the duality between differential characteristic and linear approximation, the linear approximation of the round function of the transformed cipher can be expressed as follows using the concatenation rules [4,14].

$$\Gamma V^{(i)} = \Gamma Z^{(i-1)} \oplus \Gamma Z^{(i+1)} = P^*(\Gamma X^{(i)})$$

By the way, since S -function is bijective, $H_w(\Gamma X) = H_w(\Gamma Z)$ because Γx_i is a non-zero input mask value of a linear active s_i -box. Therefore, the linear branch number \mathcal{P}_l is redefined as:

$$\mathcal{P}_l = \min_{\Gamma X \neq 0} (H_w(P^*(\Gamma X)) + H_w(\Gamma X)) = \min_{\Gamma Z \neq 0} (H_w(\Gamma V) + H_w(\Gamma Z))$$

Accordingly, if $\Delta X^{(i)}$ and $\Delta Y^{(i)}$ are exchanged for $\Gamma Z^{(i)}$ and $\Gamma V^{(i)}$, respectively, it turns out that all proofs are satisfied in the same way as for Lemma 2 to Lemma 6 and Theorem 3.

Q.E.D.

For example, the P^* -function of Camellia can be expressed as:

$$P_{Camellia}^* = \begin{bmatrix} 0 & 1 & 1 & 1 & 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 & 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 & 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 & 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 & 1 & 1 & 1 \end{bmatrix}$$

Thus, it is easily seen that $\mathcal{P}_l = 5$, and the following corollary is obtained.

Corollary 3. *Camellia with reduced to 16-rounds (without FL - and FL^{-1} -functions) has no effective linear approximation.*

Proof. The maximum linear probability of Camellia's s -boxes is $q_s = 2^{-6}$. From Theorem 5 and $\mathcal{P}_l = 5$, the maximum linear characteristic probability of Camellia with reduced to 16-rounds is also upper bounded by 2^{-132} .

Q.E.D.

6 Necessary Condition for Desirable P -Functions

In this section, we consider the necessary condition for desirable P -functions. Here, “desirable” means that the round functions are invulnerable to linear cryptanalysis as well as differential cryptanalysis.

Obviously, the condition is $\mathcal{P}_d = \mathcal{P}_l$ from Sect. 4 and Sect. 5. Thus, we investigate P -functions wherein $\mathcal{P}_d = \mathcal{P}_l$.

Theorem 6. *Assume that P -function is bijective and is expressed as an $n \times n$ matrix P over $\text{GF}(2)^m$. When the P -function satisfies $[y_i]^t = [p_{ij}][z_j]^t$, the following relations are satisfied.*

$$[\Delta y_i]^t = [p_{ij}][\Delta z_j]^t, \quad [\Gamma z_i]^t = [p_{ij}]^t[\Gamma y_j]^t = [p_{ji}][\Gamma y_j]^t,$$

where $[x_i]$ denotes the vector (or matrix) of X and $[x_i]^t$ denotes the transposed vector (or matrix) of X .

Proof. First, since $y_i = \bigoplus_{j=1}^n (p_{ij} \cdot z_j)$,

$$\begin{aligned} \Delta y_i &= y_i \oplus y'_i = \left(\bigoplus_{j=1}^n (p_{ij} \cdot z_j) \right) \oplus \left(\bigoplus_{j=1}^n (p_{ij} \cdot z'_j) \right) \\ &= \bigoplus_{j=1}^n (p_{ij} \cdot z_j \oplus p_{ij} \cdot z'_j) \\ &= \bigoplus_{j=1}^n (p_{ij} \cdot (z_j \oplus z'_j)) = \bigoplus_{j=1}^n (p_{ij} \cdot \Delta z_j) \end{aligned}$$

Thus, $[\Delta y_i]^t = [p_{ij}][\Delta z_j]^t$ is satisfied.

Second, since the P -function is bijective, $Z \cdot \Gamma Z = Y \cdot \Gamma Y$. Then,

$$\begin{aligned} Y \cdot \Gamma Y &= \bigoplus_{j=1}^n \left(\left(\bigoplus_{i=1}^n (p_{ji} \cdot z_i) \right) \cdot \Gamma y_j \right) = \bigoplus_{j=1}^n \left(\bigoplus_{i=1}^n (p_{ji} \cdot z_i \cdot \Gamma y_j) \right) \\ &= \bigoplus_{i=1}^n \left(\bigoplus_{j=1}^n ((p_{ji} \cdot \Gamma y_j) \cdot z_i) \right) = \bigoplus_{i=1}^n \left(\left(\bigoplus_{j=1}^n (p_{ji} \cdot \Gamma y_j) \right) \cdot z_i \right) \end{aligned}$$

On the other hand, since $Z \cdot \Gamma Z = \bigoplus_{i=1}^n (z_i \cdot \Gamma z_i)$, it is obvious that $\Gamma z_i = \bigoplus_{j=1}^n (p_{ji} \cdot \Gamma y_j) = \bigoplus_{j=1}^n (p_{ij}^t \cdot \Gamma y_j)$. Thus, $[\Gamma z_i]^t = [p_{ji}][\Gamma y_j]^t = [p_{ij}]^t[\Gamma y_j]^t$ is satisfied.

Q.E.D.

Theorem 7. Assume that I' is a set of matrices that consist of only one 1-element and $(m-1)$ 0-elements in each line and row, i.e., the matrices generated by only interchanging lines and/or rows of unit matrix.

If a bijective P -function can be expressed as an $n \times n$ matrix P over $\text{GF}(2)^m$ such that $P^t \cdot P \in I'$ or $P^t = I_2 \cdot P \cdot I_1$ where $I_1, I_2 \in I'$, then the P -function satisfies $\mathcal{P}_d = \mathcal{P}_l$.

Proof. By Theorem 6, if the P -function can be expressed as an $n \times n$ matrix P over $\text{GF}(2)^m$, then

$$\mathcal{P}_d = \min_{\Delta Z \neq 0} (H_w(\Delta Z) + H_w(P(\Delta Z))), \quad \mathcal{P}_l = \min_{\Gamma Y \neq 0} (H_w(P^t(\Gamma Y)) + H_w(\Gamma Y))$$

(i) In the case of $P^t \cdot P = I^* \in I'$, let $\Gamma Y = P(\Gamma W)$.

Since the P -function is bijective, it is guaranteed that $\{\Gamma Y\} = \{\Gamma W\}$. Thus,

$$\begin{aligned} \mathcal{P}_l &= \min_{\Gamma W \neq 0} (H_w(P^t(P(\Gamma W))) + H_w(P(\Gamma W))) \\ &= \min_{\Gamma W \neq 0} (H_w(I^* \cdot \Gamma W) + H_w(P(\Gamma W))). \end{aligned}$$

Here, because $I^* \in I'$, $I^* \cdot \Gamma W$ leads to another vector simply by interchanging the elements of ΓW . Thus, $H_w(\Gamma W) = H_w(I^* \cdot \Gamma W)$. As a result, $\exists(\Delta Z, \Gamma W)$, s.t. $\mathcal{P}_d = \mathcal{P}_l$.

(ii) In the case of $P^t = I_2 \cdot P \cdot I_1$ where $I_1, I_2 \in I'$, as mentioned above, since I_1 and I_2 lead to another vector simply by interchanging the elements, $H_w(\Delta X) = H_w(I_1(\Delta X))$ and $H_w(I_2(\Gamma W)) = H_w(\Gamma W)$. Now, let $\Delta Z = I_1(\Delta X)$. Since I_1 is bijective, it is guaranteed that $\{\Delta X\} = \{\Delta Z\}$. Thus,

$$\begin{aligned} \mathcal{P}_d &= \min_{\Delta X \neq 0} (H_w(I_1(\Delta X)) + H_w(P \cdot I_1(\Delta X))) \\ &= \min_{\Delta X \neq 0} (H_w(\Delta X) + H_w(P \cdot I_1(\Delta X))). \end{aligned}$$

On the other hand,

$$\begin{aligned} \mathcal{P}_l &= \min_{\Gamma Y \neq 0} (H_w(I_2 \cdot P \cdot I_1(\Gamma Y)) + H_w(\Gamma Y)) \\ &= \min_{\Gamma Y \neq 0} (H_w(P \cdot I_1(\Gamma Y)) + H_w(\Gamma Y)). \end{aligned}$$

As a result, $\exists(\Delta X, \Gamma Y)$, s.t. $\mathcal{P}_d = \mathcal{P}_l$.

Q.E.D.

For example, the relationship between P -function and P^* -function of Camellia is shown as follows. Thus Theorem 7 indicates that the P -function of Camellia is “desirable.”

$$P_{\text{Camellia}}^* = P_{\text{Camellia}}^t = I^* \cdot P_{\text{Camellia}} \cdot I^*,$$

because

$$P_{Camellia} = \begin{bmatrix} 0 & 1 & 1 & 1 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 1 & 1 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 & 1 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 & 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 & 1 & 1 & 0 & 1 \end{bmatrix}, \quad P_{Camellia}^* = \begin{bmatrix} 0 & 1 & 1 & 1 & 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 & 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 & 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 & 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 & 1 & 1 & 1 \end{bmatrix}, \quad I^* = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \end{bmatrix}$$

7 Conclusion

This paper studied the upper bounds of the maximum differential and linear characteristic probabilities of Feistel ciphers with SPN round function. In the same way as for SPN ciphers, we considered the minimum number of differential and linear active s -boxes, which are a measure of the upper bounds of these probabilities, in order to evaluate security against differential and linear cryptanalyses. The advantage of this method is that it considers the interrelation between input and output data in consecutive rounds, unlike Knudsen's estimation.

We focused on the minimum number of active s -boxes in some consecutive rounds of Feistel ciphers, i.e., in three, four, six, eight, and twelve consecutive rounds, since they can determine the upper bounds of the maximum differential and linear probabilities using the differential and linear branch numbers \mathcal{P}_d , \mathcal{P}_l , respectively. These numbers provide the avalanche effects of P -functions with regard to differential and linear characteristics. As a result, we clarified that the lower bounds of the minimum number of differential (resp. linear) active s -boxes are 2, \mathcal{P}_d (\mathcal{P}_l), $\mathcal{P}_d + 2$ ($\mathcal{P}_l + 2$), $2\mathcal{P}_d + 1$ ($2\mathcal{P}_l + 1$), and $3\mathcal{P}_d + 1$ ($3\mathcal{P}_l + 1$), respectively. The interesting result is that the lower bound of the minimum number of active s -boxes is proportional to the branch number every fourth round, while it seems to be every third round at first glance. Furthermore, this means that, if the branch number is the same, a $2r$ -round Feistel cipher has almost same invulnerability to differential and linear cryptanalyses as a r -round SPN cipher in terms of the upper bounds of the maximum differential and linear probabilities.

Finally, we investigated the necessary condition for desirable P -functions, which means that the round functions are invulnerable to both differential and linear cryptanalyses. In addition, we showed the example of the round function of Camellia, which satisfies the condition.

References

1. K. Aoki, T. Ichikawa, M. Kanda, M. Matsui, S. Moriai, J. Nakajima, and T. Tokita, "Camellia: A 128-bit Block Cipher Suitable for Multiple Platforms – Design and Analysis –," *Selected Areas in Cryptography — 7th Annual International Workshop, SAC2000*, LNCS in this proceeding.

2. K. Aoki, K. Kobayashi, and S. Moriai, "Best Differential Characteristic Search of FEAL," *Fast Software Encryption — 4th International Workshop, FSE'97*, LNCS **1267**, pp.41–53, 1997.
3. K. Aoki, and K. Ohta, "Strict Evaluation of the Maximum Average of Differential Probability and the Maximum Average of Linear Probability," *IEICE Transactions Fundamentals of Electronics, Communications and Computer Sciences*, Vol. E80-A, No. 1, pp. 2–8, 1997.
4. E. Biham, "On Matsui's Linear Cryptanalysis," *Advances in Cryptology — EUROCRYPT'94*, LNCS **950**, pp.341–355, 1995.
5. E. Biham, R. Anderson, and L. R. Knudsen, "Serpent: A New Block Cipher Proposal," *Fast Software Encryption — 5th International Workshop, FSE'98*, LNCS **1372**, pp.222–238, 1998.
6. E. Biham, and A. Shamir, "Differential cryptanalysis of DES-like cryptosystems," *Journal of Cryptology*, Vol.4, No.1, pp.3–72, 1991.
7. Data Encryption Standard, FIPS-PUB-46, 1977.
8. J. Daemen, L. Knudsen, and V. Rijmen, "The block cipher SQUARE," *Fast Software Encryption — 4th International Workshop, FSE'97*, LNCS **1267**, pp.54–68, 1997.
9. S. Hong, S. Lee, J. Lim, J. Sung, and D. Cheon, "Provable Security against Differential and Linear Cryptanalysis for the SPN structure," *Fast Software Encryption Workshop 2000*, 2000. (LNCS to appear).
10. L. R. Knudsen, "Practically Secure Feistel Ciphers," *Fast Software Encryption — Cambridge Security Workshop*, LNCS **809**, pp.211–221, 1994.
11. M. Kanda, Y. Takashima, T. Matsumoto, K. Aoki, and K. Ohta, "A strategy for constructing fast round functions with practical security against differential and linear cryptanalysis," *Selected Areas in Cryptography — 5th Annual International Workshop, SAC'98*, LNCS **1556**, pp.264–279, 1999.
12. X. Lai, J. L. Massey, and S. Murphy, "Markov ciphers and differential cryptanalysis," *Advances in Cryptology — EUROCRYPT'91*, LNCS **547**, pp.17–38, 1991.
13. M. Matsui, "Linear cryptanalysis method for DES cipher," *Advances in Cryptology — EUROCRYPT'93*, LNCS **765**, pp.386–397, 1994.
14. M. Matsui, "On Correlation Between the Order of S-boxes and the Strength of DES," *Advances in Cryptology — EUROCRYPT'94*, LNCS **950**, pp.366–375, 1995.
15. M. Matsui, "New Block Encryption Algorithm MISTY," *Fast Software Encryption — 4th International Workshop, FSE'97*, LNCS **1267**, pp.54–68, 1997.
16. S. Moriai, K. Aoki, and K. Ohta, "The Best Linear Expression Search of FEAL," *IEICE Transactions Fundamentals of Electronics, Communications and Computer Sciences*, Vol. E79-A, No. 1, pp. 2–11, 1996.
17. K. Nyberg, "Linear Approximation of Block Ciphers," *Advances in Cryptology — EUROCRYPT'94*, LNCS **950**, pp.439–444, 1995.
18. K. Nyberg, and L. R. Knudsen, "Provable Security Against a Differential Attack," *Journal of Cryptology*, Vol. 8 No. 1, pp. 27–37, 1995.
19. V. Rijmen, J. Daemen, B. Preneel, A. Bosselaers, and E. D. Win, "The cipher SHARK," *Fast Software Encryption — Third International Workshop*, LNCS **1039**, pp.99–111, 1996.

Differential Cryptanalysis of Reduced Rounds of GOST

Haruki Seki¹ and Toshinobu Kaneko²

¹ TAO (Telecommunications Advancement Organization of Japan), 1-1-32
Shin'urashima-cho, Kanagawa-ku, Yokohama, 221-0031 Japan
`hseki@yokohama.tao.go.jp`

² Science University of Tokyo,
2641 Yamazaki, Noda-shi, Chiba, 278-8510 Japan
`kaneko@ee.noda.sut.ac.jp`

Abstract. The block cipher GOST was proposed in former Soviet Union in 1989. In this paper we present the first result of differential cryptanalysis of GOST with reduced number of rounds. By introducing the idea of using a set of differential characteristics, which is a partitioning type, we can reduce the influence of the key value upon the probability as well as get high differential probability. Using 2^{51} chosen plaintexts the key of 13-round GOST can be obtained. Next this differential cryptanalysis is expanded with combining related-key attack. Using 2^{56} chosen plaintexts the key of 21 rounds of GOST can be obtained.

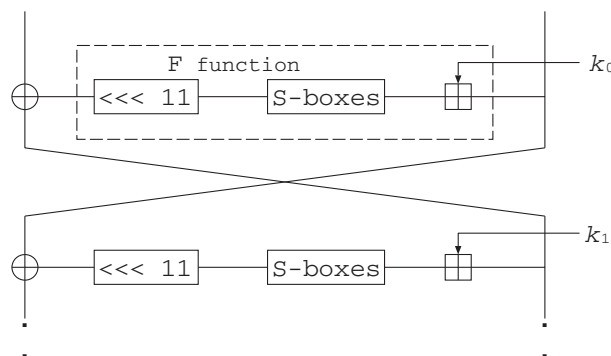
1 Introduction

The block cipher GOST was proposed in former Soviet Union in 1989[1]. GOST is an acronym for “Gosudarstvennyi Standard”, or Government Standard.

In this paper we present the first result of differential cryptanalysis of GOST with reduced number of rounds. Next the analysis is expanded with combining related-key attack.

GOST has key addition modulo 2^{32} in each round function. So orthodox differential cryptanalysis using one characteristic is not useful. The reason is that the probability of differential characteristic varies with not only the value of input-output difference but the value of the sub-key, frequently become zero. To overcome this we introduce the idea of using a set of differential characteristics. This is similar to truncated differential attack[2,3,4] in predicting only parts of all output bit value. But it is slightly different in the sense that this attack uses a set of differentials of S-boxes and applies this to round function, which is a partitioning type, and construct a new type of 2-round iterative characteristic. By this characteristic we can reduce the influence of the key value upon the probability as well as get high differential probability. On average using 2^{51} chosen plaintexts the key of 13-round GOST can be obtained. In the case of keys which make the probability the highest, 17 rounds of GOST can be attacked.

This differential cryptanalysis is expanded with combining related-key attack[5].
John Kelsey et al applied related-key attack to GOST[6]. But no concrete characteristics was revealed in [6]. In this paper we show the concrete characteristics.

**Fig. 1.** GOST round function

On average using 2^{56} chosen plaintexts the key of 21-round GOST can be obtained.

These attacks are also applicable, even if the S-boxes are randomly generated.

This paper is organized as follows. Section 2 briefly reviews algorithm of GOST. In Section 3 we describe a differential cryptanalysis of GOST using one differential characteristic. In Section 4 we describe a differential cryptanalysis of GOST using a set of differential characteristics. In Section 5 we discuss the differential cryptanalysis with combining related-key attack. In Section 6 we discuss the differential cryptanalysis in the case of random S-boxes. We conclude in Section 7.

2 Description of GOST

The block cipher GOST is based on the framework of the Feistel cipher. GOST has 32 rounds, 64-bit blocksize, and 256-bit keysize. The F -function consists of operations specified as follows(see also Figure 1).

$+$: Addition modulo 2^{32}

S-boxes : 8 different 4×4 -bit S-boxes S_1, S_2, \dots, S_8

$\lll 11$: 11-bit left rotation

The S-boxes are not specified in the standard. In this paper we use a set of S-boxes used in an application for the Central Bank of the Russian Federation (tables of S-boxes are given in page 333 of [7], see also Appendix A.).

Key-schedule is simple. The 256-bit master-key is divided to eight 32-bit blocks : k_1, k_2, \dots, k_8 . Each round uses the subkey as shown in Table below.

| | | | | | | | | | | | | | | | | | | | | | | | | |
|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|------------|-------|-------|-------|------------|-------|-------|-------|-------|-------|-------|-------|-------|-------|
| Round | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | ~ 15 | 16 | 17 | 18 | ~ 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 | 32 |
| key | k_1 | k_2 | k_3 | k_4 | k_5 | k_6 | k_7 | k_8 | k_1 | k_2 | $\sim k_7$ | k_8 | k_1 | k_2 | $\sim k_7$ | k_8 | k_8 | k_7 | k_6 | k_5 | k_4 | k_3 | k_2 | k_1 |

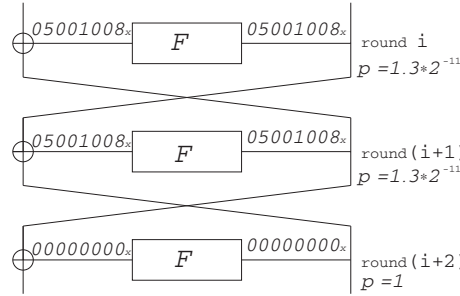


Fig. 2. One of the best 3-round iterative characteristic

3 An Attack Using One Differential Characteristic

GOST has key addition modulo 2^{32} in each round. In such a cipher the differential probability varies with not only the value of input-output difference but the value of the sub-key, and frequently become zero (see also Appendix B). Figure 2 shows one of the best 3-round iterative characteristic in case of S-boxes used in an application for the Central Bank of the Russian Federation[7]¹. This characteristic has the probability described below in one round.

$$0 \leq \text{Prob}\{05001008_x \rightarrow 05001008_x\} \leq 1.5 \times 2^{-7} \quad (1)$$

Where $X \rightarrow Y$ means that an *inputxor* X result in an *outputxor* Y. Average probability over all key values is 1.3×2^{-11} in one round. 8-round characteristic has probability 2^{-53} . So using 2-Round attack 10-round GOST is expected to be attacked using 2^{56} chosen plaintexts. But more than half of sub-key space makes the differential probability of each S-box to be zero (See also Appendix B). In 8-round characteristic the chance for the probability to be nonzero is only 2×10^{-5} . Consequently attack using one differential characteristic is not useful for GOST.

4 Cryptanalysis of GOST Using a Set of Differential Characteristics

To overcome the dependence of differential probability on the key, we introduce the idea of using a set of differential characteristics. This is slightly different from truncated differentials in the sense that this attack use a set of differentials of S-boxes and apply this to round function, which is a partitioning type, and construct a new type of 2-round iterative characteristics. By this characteristics we can considerably reduce the influence of the key value as well as get higher differential probability than described in Section 3.

¹ A 3-round iterative characteristic which has 2 active S-boxes in each round is impossible

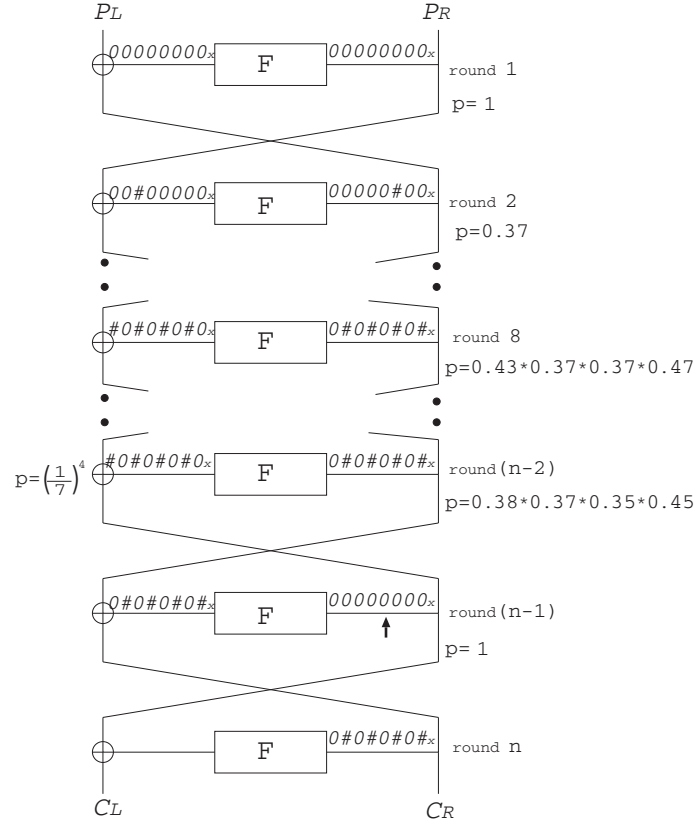


Fig. 3. A set of differential characteristics

4.1 A Set of Differential Characteristics

We use a set of differential characteristics as shown in Figure 3. The differences of plaintext pairs are $(00000\#00_x \parallel 00000000_x)$. $\#$ means nonzero 4-bit difference whose MSB (most significant bit) is zero.

This set of differential characteristics are possible when LSB (least significant bit) of output difference of each active S-box is zero. The number of active S-boxes increases one by one with the number of rounds, and saturates with 4 after round 8.

At first we estimate the probability of differentials of each S-box. That is $\text{Prob}\{\text{nonzero difference whose MSB is } 0 \rightarrow \text{nonzero difference whose LSB is } 0\}$. The probability varies from 0.30 to 0.75 depending on S-box number and the key values (see Appendix C for details). Let p_{S_i} be average probability of S_i for all key values. The average probability of each round is the products of p_{S_i} of active S-boxes. For example the average probability of round 8, 10, and so on is $p_{S_1} \times p_{S_3} \times p_{S_5} \times p_{S_7} = .43 \times .37 \times .37 \times .47$.

4.2 An Attack on GOST

To recover the last round sub-key, we use the characteristics as shown in Figure 3. At first we have to fix all 32 bits of input difference to $(n-1)$ -th F -function to zero. To realize this a cancellation has to happen at an xor operation after the $(n-2)$ -th F -function. If each value of \sharp is randomly distributed from 1 to 7, the probability of getting zero difference is $(\frac{1}{7})^4$. We estimate the probability p_n of this characteristics for n -round GOST. The probability is shown as follows when n is even².

$$p_n = p_{S_3}^{\frac{n}{2}-1} \times p_{S_6}^{\frac{n}{2}-2} \times p_{S_1}^{\frac{n}{2}-2} \times p_{S_4}^{\frac{n}{2}-3} \times p_{S_7}^{\frac{n}{2}-3} \\ \times p_{S_2}^{\frac{n}{2}-4} \times p_{S_5}^{\frac{n}{2}-4} \times p_{S_8}^{\frac{n}{2}-5} \times \left(\frac{1}{7}\right)^4. \quad (2)$$

S/N-ratio is defined as follows[8].

$$S/N = \frac{2^k \times p}{\alpha \times \beta}.$$

k = number of key bits we are looking for

p = probability of characteristics

α = average count of keys per analyzed pair

β = ratio of analyzed pairs to all pairs

In this 2-Round attack each value is described as follows³.

$$k = 32, \quad \alpha = 1, \quad \beta = 2^{-20}$$

Consequently we get

$$S/N = p_n \times 2^{52}. \quad (3)$$

Table 1 shows the estimated values of p , S/N , and the number of chosen plaintexts needed. If we choose a structure of 2^3 plaintexts which differ only at 3 bits of $P_L(\sharp$ in Figure 3), one structure proposes 28 pairs of plaintexts. For example 2^{45} plaintexts propose about 2^{47} pairs. In the case of the keys which make the probability of differential characteristics the highest, p_{17} equals 1.6×2^{-49} and 17-round GOST can be attacked.

5 A Related-Key Attack

A related-key attack were first described in [5]. John Kelsey et al. proposed related-key attack of GOST[6]. But no concrete characteristics was revealed. In

² When n is odd the probability is shown in a similar way

³ 20 bits of C_R are fixed to zero, so $\beta = 2^{-20}$. All bits of $C_L \oplus F_n(C_R)$ are fixed to zero, so $\alpha = 1$

Table 1. Estimates of pairs needed for differential attack

| Rounds | Prob. | S/N | Chosen Plaintexts |
|--------|----------------------|-------|-------------------|
| 12 | 1.2×2^{-44} | 2^8 | 2^{45} |
| 13 | 1.7×2^{-50} | 7 | 2^{51} |
| 14 | 1.5×2^{-55} | 0.4 | impossible |

Table 2. Estimates of pairs needed for related-key attack

| Rounds | Prob. | S/N | Chosen Plaintexts |
|--------|----------------------|------------------|-------------------|
| 20 | 1.8×2^{-47} | 1.8×2^5 | 2^{49} |
| 21 | 1.3×2^{-52} | 1.3 | 2^{56} |
| 22 | 1.1×2^{-57} | 2^{-5} | impossible |

this section the differential cryptanalysis mentioned in Section 4.2 is expanded with combining related-key attack, and the concrete characteristics are shown. Two unknown related keys K and K^* are used for attack. The relationship between two keys are described as follows.

$$K = (k_1, k_2, \dots, k_8)$$

$$K^* = (k_1 \oplus 80000000_x, k_2, \dots, k_8)$$

Using plaintext $P = (P_L, P_R)$ for key K and $P^* = (P_L \oplus 00000700_x, P_R)$ for key K^* , we can bypass the first 8 rounds for free with probability $\frac{1}{4}$ ⁴. Figure 4 shows the differential characteristics using related-key attack.

In round 9 output difference is $00000\#00_x$ with probability $\frac{3}{4}$. After round 10 the differential characteristics are the same as described in Section 4.2. Consequently the probability of the differential characteristics of n -round GOST is $\frac{1}{4} \times \frac{3}{4} \times p_{n-8}$. Where p_{n-8} is calculated from equation (2).

Table 2 shows the estimated values of p , S/N, and the number of chosen plaintexts needed for attack.

6 In the Case of Random S-Boxes

The S-boxes are not specified in GOST. In this section we discuss the attack in the case of random S-boxes. We have generated 100,000 of random S-boxes, and obtained the $Prob\{\text{nonzero difference whose MSB is } 0 \rightarrow \text{nonzero difference whose LSB is } 0\}$. Table 3 shows the probability, corresponding ratio of the number of S-boxes, and the number of rounds we can attack. On average 12 rounds of GOST can be attacked with a set of differential characteristics.

Next we consider the case of the analysis with combining related-key attack. The best characteristic in round 1 to bypass the first 8 rounds for free varies from

⁴ $(k_1 \oplus 80000000_x) + P_R = k_1 + (P_R \oplus 80000000_x)$. So this probability is equal to $Prob\{8_x \rightarrow e_x\}$. For all values of k_1 this probability of S_8 is $\frac{1}{4}$

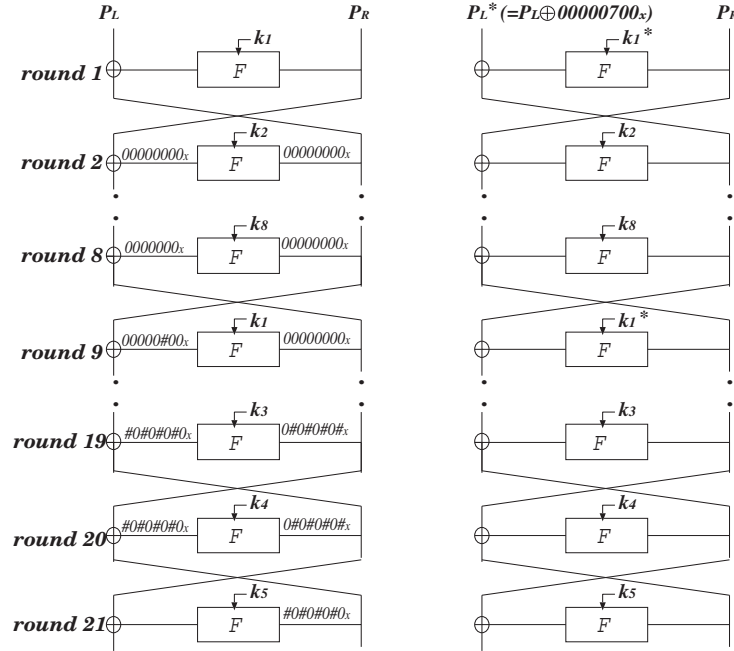


Fig. 4. The differential characteristics with related key

Table 3. Estimates in the case of random S-boxes

| Prob | $0.34 \leq$ | $0.38 \leq$ | $0.42 \leq$ | $0.47 \leq$ | $0.50 \leq$ | $0.54 \leq$ | 0.625 |
|-----------------------|-------------|-------------|-------------|-------------|-------------|-------------|---------|
| Ratio(%) | 62 | 27 | 8 | 2.3 | 0.5 | 0.4 | |
| Rounds (differential) | 12 | 13 | 14 | 15 | 16 | 17 ~ 20 | 20 |
| Rounds (related-key) | 19 | 20 | 21 | 22 | 23 | 24 ~ 26 | 27 |

the S-box construction. But we can always find the characteristic which has the probability larger than $\frac{1}{8}$. So we use this probability for every S-boxes here. On average 19 rounds of GOST can be attacked with combining related-key attack.

The maximum probability of all random S-boxes⁵ is 0.625. In this case 20 rounds of GOST can be attacked using a set of differential characteristics, and 27 rounds of GOST can be attacked with combining related-key attack.

Consequently this set of differential characteristics is useful even if S-boxes are randomly generated.

7 Conclusion

In this paper we described the first result of an attack on GOST with reduced number of rounds using a set of differential characteristics, which is a partitioning

⁵ The S-box which has the maximum probability is $\{9, 7, 5, 1, 11, 15, 3, 13, 0, 4, 12, 10, 14, 8, 2, 6\}$

type. In the case of S-boxes used in an application for the Central Bank of the Russian Federation, on average using 2^{51} chosen plaintexts the key of 13-round GOST can be obtained. In the case of the keys which make the probability of differential characteristics the highest, 17-round GOST can be attacked. The analysis is also expanded with combining related-key attack. Using 2^{56} chosen plaintexts the key of 21-round GOST can be obtained.

We also show this attack is applicable, even if the S-boxes are randomly generated. On average 12 rounds of GOST can be attacked with a set of differential characteristics, and 19 rounds of GOST can be attacked with combining related-key attack. In the case of the weakest S-box 20 rounds of GOST can be attacked with a set of differential characteristics, and 27 rounds of GOST can be attacked with combining related-key attack.

Appendix A: S-Boxes

A set of S-boxes used in an application for the Central Bank of the Russian Federation are given in page 333 of [7].

$$\begin{aligned}
S_8 &= \{1, 15, 13, 0, 5, 7, 10, 4, 9, 2, 3, 14, 6, 11, 8, 12\} \\
S_7 &= \{13, 11, 4, 1, 3, 15, 5, 9, 0, 10, 14, 7, 6, 8, 2, 12\} \\
S_6 &= \{4, 11, 10, 0, 7, 2, 1, 13, 3, 6, 8, 5, 9, 12, 15, 14\} \\
S_5 &= \{6, 12, 7, 1, 5, 15, 13, 8, 4, 10, 9, 14, 0, 3, 11, 2\} \\
S_4 &= \{7, 13, 10, 1, 0, 8, 9, 15, 14, 4, 6, 12, 11, 2, 5, 3\} \\
S_3 &= \{5, 8, 1, 13, 10, 3, 4, 2, 14, 15, 12, 7, 6, 0, 9, 11\} \\
S_2 &= \{14, 11, 4, 12, 6, 13, 15, 10, 2, 3, 8, 1, 0, 7, 5, 9\} \\
S_1 &= \{4, 10, 9, 2, 13, 8, 0, 14, 6, 11, 1, 12, 7, 15, 5, 3\}
\end{aligned}$$

Appendix B: An Example of Dependence of Differential Probability on the Key

The table below shows the $Prob\{8_x \rightarrow 2_x\}$ of S_1 , $Prob\{1_x \rightarrow a_x\}$ of S_4 , $Prob\{5_x \rightarrow 1_x\}$ of S_7 used in an application for the Central Bank of the Russian Federation. The key values in the table are 4 bits of the round-key whose bit positions are corresponding to each S-box. We don't count the case in which differential carry bit occurs to the fourth position, because carry bit doesn't hold the input difference of upper S-box to be zero. This table shows that the probability becomes zero for more than half of the key space.

| key | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | a | b | c | d | e | f |
|-------|-----|-----|-----|-----|-----|-----|-----|---|-----|-----|-----|-----|-----|-----|-----|-----|
| S_7 | 0 | .13 | 0 | .25 | 0 | .13 | 0 | 0 | 0 | .13 | 0 | .25 | 0 | .13 | 0 | 0 |
| S_4 | .38 | 0 | .38 | 0 | .38 | 0 | .38 | 0 | .38 | 0 | .38 | 0 | .38 | 0 | .38 | 0 |
| S_1 | .13 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | .13 | .13 | .13 | .13 | .13 | .13 | .13 |

Appendix C: Differential Distribution Table of Each S-Box

This table shows $Prob\{\text{nonzero difference whose MSB is } 0 \rightarrow \text{nonzero difference whose LSB is } 0\}$ of each S-box used in an application for the Central Bank of the Russian Federation. We don't count the case in which differential carry bit to the fourth position occurs, because carry bit doesn't hold the input difference of upper S-box to be zero.

| key | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | a | b | c | d | e | f | average p_{S_i} |
|-------|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-------------------|
| S_8 | .46 | .46 | .43 | .43 | .43 | .46 | .46 | .46 | .46 | .46 | .43 | .43 | .43 | .46 | .46 | .46 | .45 |
| S_7 | .75 | .55 | .43 | .36 | .39 | .38 | .43 | .55 | .75 | .55 | .43 | .36 | .39 | .38 | .43 | .55 | .47 |
| S_6 | .43 | .39 | .32 | .30 | .32 | .32 | .36 | .39 | .43 | .39 | .32 | .30 | .32 | .32 | .36 | .39 | .35 |
| S_5 | .46 | .39 | .36 | .32 | .32 | .32 | .36 | .43 | .46 | .39 | .36 | .32 | .32 | .32 | .36 | .43 | .37 |
| S_4 | .46 | .38 | .36 | .32 | .39 | .32 | .36 | .39 | .46 | .38 | .36 | .32 | .39 | .32 | .36 | .39 | .37 |
| S_3 | .43 | .39 | .32 | .32 | .32 | .32 | .43 | .39 | .43 | .39 | .32 | .32 | .32 | .32 | .43 | .39 | .37 |
| S_2 | .46 | .43 | .36 | .36 | .32 | .38 | .36 | .38 | .46 | .43 | .36 | .36 | .32 | .38 | .36 | .38 | .38 |
| S_1 | .57 | .55 | .43 | .36 | .39 | .36 | .36 | .43 | .57 | .55 | .43 | .36 | .39 | .36 | .36 | .43 | .43 |

References

1. GOST, Gosudarstvennyi Standard 28147-89, "Cryptographic Protection for Data Processing Systems", Government Committee of the USSR for Standards, 1989.
2. L.R.Knudsen, "Truncated and higher order differentials", FSE'94, Lecture Notes in Computer Science, pp.196-211, Springer-Verlag, 1994.
3. L.R.Knudsen,T.A.Berson, "Truncated Differentials of SAFER", FSE'96, Lecture Notes in Computer Science, pp.15-26, Springer-Verlag, 1996.
4. J.Borst,L.R.Knudsen,V.Rijmen, "Two Attacks on Reduced IDEA", Eurocrypt'97, Lecture Notes in Computer Science, pp.1-13, Springer-Verlag, 1997.
5. E.Biham, "New Types of Cryptanalytic Attacks Using Related Keys", Eurocrypt'93, Lecture Notes in Computer Science, pp.398-409, Springer-Verlag, 1993.
6. J.Kelsey, B.Shneier, D.Wagner, "Key-Schedule Cryptanalysis of IDEA, G-DES, GOST, SAFER, and Triple-DES", CRYPTO'96 Proceedings, Springer-Verlag, 1996, pp.237- 251.
7. B.Shneier, "Applied Cryptography", John Wiley & Sons, pp. 331-334.
8. E.Biham, A.Shamir., "Differential Cryptanalysis of DES-like Cryptosystems," Journal of Cryptology 1991.

Practical Security Evaluation against Differential and Linear Cryptanalyses for Feistel Ciphers with SPN Round Function

Masayuki Kanda

NTT Information Sharing Platform Laboratories,
1-1-612A Hikarinooka, Yokosuka-shi, Kanagawa, 239-0847, Japan
kanda@isl.ntt.co.jp

Abstract. This paper studies the upper bounds of the maximum differential and linear characteristic probabilities of Feistel ciphers with SPN round function. In the same way as for SPN ciphers, we consider the minimum number of differential and linear active s -boxes, which provides a measure of the upper bounds of these probabilities, in order to evaluate the security against differential and linear cryptanalyses. The purpose of this work is to clarify the (lower bound of) minimum numbers of active s -boxes in some consecutive rounds of Feistel ciphers, i.e., in three, four, six, eight, and twelve consecutive rounds, using differential and linear branch numbers \mathcal{P}_d , \mathcal{P}_l , respectively. Furthermore, we investigate the necessary condition for desirable P -functions, which means that the round functions are invulnerable to both differential and linear cryptanalyses. As an example, we show the round function of Camellia, which satisfies the condition.

1 Introduction and Motivation

The best known attacks are differential cryptanalysis [6] proposed by Biham and Shamir and linear cryptanalysis [13] proposed by Matsui. Since these cryptanalyses are the most powerful approaches known for attacking many symmetric block ciphers, designers should evaluate the security of any new proposed ciphers against differential and linear cryptanalyses. To do this it is necessary to determine the maximum differential and linear probabilities by a useful (and acceptable) method. Feistel ciphers are commonly analyzed by (a) the upper bounds of the maximum average of differential and linear hull probabilities or (b) the maximum differential and linear characteristic probabilities. SPN ciphers, on the other hand, are commonly analyzed by (c) the upper bounds of the maximum differential and linear characteristic probabilities. Recently, Hong et al. showed (a) the upper bounds of the maximum average of differential and linear hull probabilities of SPN ciphers [9].

With reference to method (a), Nyberg and Knudsen showed that the maximum average of differential and linear hull probabilities for r -round ($r \geq 4$) Feistel ciphers are bounded by $2p^2$, $2q^2$ if the maximum differential and linear

probabilities of the round function are p , q , respectively¹ [18]. They stated that Feistel ciphers are *provably secure* against differential and linear cryptanalyses if these probabilities are sufficiently low. This means that they are theoretically invulnerable to differential and linear cryptanalyses, since these probabilities are the upper bounds of the average of differential and linear hull probabilities. However, this approach has one fatal disadvantage. That is, these probabilities settle at some constant value even if the number of rounds increases. Therefore, a round function has to yield extremely low maximum differential and linear probabilities. This imposes a hard restriction on designing the round function. As a matter of fact, for a commercial cipher, MISTY [15] is provably secure with respect to differential and linear cryptanalyses.

Method (b) has been used to estimate many (extended) Feistel ciphers such as DES [6,13] and FEAL [16,2]. Biham and Shamir claimed that the higher the differential characteristic probability is, the higher the success rate of differential cryptanalysis is. This is because they exploited a single path between plaintexts and ciphertexts which holds significant differential characteristic probability. Matsui also claimed the same for linear cryptanalysis. Thus, Feistel ciphers are *sufficiently secure* against differential and linear cryptanalyses if these probabilities are less than the security threshold. Strictly speaking, however, these probabilities only give the lower bounds of the maximum average of differential and linear hull probabilities, since this method does not consider multiple paths between the same plaintexts and ciphertexts [12,17].

For SPN ciphers, Rijmen et al. introduced the branch number \mathcal{B} [19]. The number \mathcal{B} is the minimum number of active s -boxes in two consecutive rounds of a non-trivial differential characteristic or a non-trivial linear trail. Since each active s -box reduces the differential and linear characteristic probabilities, the number \mathcal{B} provides the upper bounds of the maximum differential and linear characteristic probabilities in two consecutive rounds. The security against differential and linear cryptanalyses is evaluated by piling up the number \mathcal{B} every two rounds. It is noted that Knudsen proposed a very similar concept for Feistel ciphers [10]. He noted that Feistel ciphers are *practically secure* against differential and linear cryptanalyses if the upper bounds of the maximum differential and linear characteristic probabilities are less than the security threshold.

It is obvious that the upper bounds of the maximum differential and linear characteristic probabilities by method (c) lie between the upper bounds of the maximum average of differential and linear hull probabilities by method (a) and the maximum differential and linear characteristic probabilities by method (b). Moreover, for most ciphers, the maximum averages of differential and linear hull probabilities, which provide the actual invulnerability to differential and linear cryptanalyses, are much lower than the upper bounds of these probabilities if the number of rounds increases. Therefore, it is worth investigating the upper bounds of the maximum differential and linear characteristic probabilities.

¹ Aoki and Ohta showed that these probabilities are bounded by p^2 , q^2 if the round function is bijective and $r \geq 3$ [3]

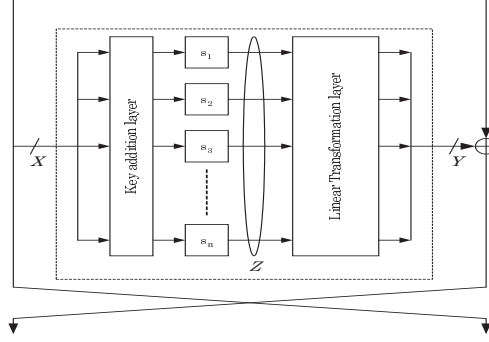


Fig. 1. SPN round function

since we assume that the round key, which is used within one round, consists of independent and uniformly random bits, and is bitwise XORed with data.

Notations describe the model as below.

S -function is a non-linear transformation layer with n parallel m -bit bijective s -boxes. That is,

$$\begin{aligned} S : (\mathbb{Z}_2^m)^n &\longrightarrow (\mathbb{Z}_2^m)^n \\ X = (x_1, \dots, x_n) &\mapsto Z = S(X) = (s_1(x_1), \dots, s_n(x_n)) \end{aligned}$$

P -function is a linear transformation layer, i.e.,

$$\begin{aligned} P : (\mathbb{Z}_2^m)^n &\longrightarrow (\mathbb{Z}_2^m)^n \\ Z = (z_1, \dots, z_n) &\mapsto Y = P(Z) = (y_1, \dots, y_n) \end{aligned}$$

Finally, the SPN round function can be described as follows.

$$\begin{aligned} F : (\mathbb{Z}_2^m)^n &\longrightarrow (\mathbb{Z}_2^m)^n \\ X = (x_1, \dots, x_n) &\mapsto Y = F(X) = P(S(X)) = (y_1, \dots, y_n) \end{aligned}$$

Let $X^{(i)}$ be the input data to the i -th round function, and $Y^{(i)}$ be the i -round output data. The Feistel cipher is defined as:

$$X^{(i+1)} = X^{(i-1)} \oplus Y^{(i)} \quad (1 \leq i \leq r),$$

where $(X^{(1)}|X^{(0)})$ is a plaintext and $(X^{(r)}|X^{(r+1)})$ is a ciphertext.

2.3 Definitions

We use the following definitions in this paper.

Definition 1. For any given $\Delta x, \Delta z, \Gamma x, \Gamma z \in \mathbb{Z}_2^m$, the differential and linear probabilities of each s -box s_i are defined as:

$$\begin{aligned} DP^{s_i}(\Delta x \rightarrow \Delta z) &= \frac{\#\{x \in \mathbb{Z}_2^m | s_i(x) \oplus s_i(x \oplus \Delta x) = \Delta z\}}{2^m} \\ LP^{s_i}(\Gamma z \rightarrow \Gamma x) &= \left(2 \times \frac{\#\{x \in \mathbb{Z}_2^m | x \cdot \Gamma x = s_i(x) \cdot \Gamma z\}}{2^m} - 1 \right)^2 \end{aligned}$$

Definition 2. The maximum differential and linear probabilities of s -boxes are defined as:

$$p_s = \max_i \max_{\Delta x \neq 0, \Delta z} DP^{s_i}(\Delta x \rightarrow \Delta z)$$

$$q_s = \max_i \max_{\Gamma x, \Gamma z \neq 0} LP^{s_i}(\Gamma z \rightarrow \Gamma x)$$

This means that p_s, q_s are the upper bounds of the maximum differential and linear probabilities for all s -boxes.

Definition 3. A differential active s -box is defined as an s -box given a non-zero input difference, while a linear active s -box is defined as an s -box given a non-zero output mask value [11].

Note: When an s -box is bijective, s -boxes given a non-zero output difference and a non-zero input mask value are also differential and linear active s -boxes, respectively.

Definition 4. Let $X = (x_1, \dots, x_n) \in \text{GF}(2^m)^n$ then the Hamming weight of X is denoted by

$$H_w(X) = \#\{i | x_i \neq 0\}.$$

This means that the Hamming weight of X equals the number of non-zero m -bit characters from $\text{GF}(2^m)$ of X .

3 Previous Works – the Security of SPN Ciphers

As mentioned above, the security of most SPN ciphers against differential and linear cryptanalyses is evaluated using the (lower bound of) minimum number of differential and linear active s -boxes, which are a measure of the upper bounds of differential and linear characteristic probabilities [19,8,5]. To determine the (lower bound of) minimum number of active s -boxes, Rijmen et al. defined the branch number \mathcal{B} [19].

Definition 5. In SPN ciphers, the differential branch number \mathcal{B}_d is defined as:

$$\mathcal{B}_d = \min_{\Delta X \neq 0} (H_w(\Delta X) + H_w(\theta(\Delta X))),$$

where ΔX is an input difference into the diffusion layer and $\theta(\Delta X)$ is an output difference from the layer.

Note that ΔX is also an output difference from a substitution layer and $\theta(\Delta X)$ is also an input difference to the next substitution layer. Since s -boxes are bijective, $H_w(\Delta X)$ equals the number of differential active s -boxes in the substitution layer and $H_w(\theta(\Delta X))$ equals that in the next substitution layer. That is, if n_d is the minimum number of differential active s -boxes in two consecutive rounds, then $n_d = \mathcal{B}_d$. Thus, it turns out that the minimum number of differential active s -boxes in $2r$ -round SPN ciphers is lower bounded by $r\mathcal{B}_d$, and the following theorem is obtained.

Theorem 1. *The maximum differential characteristic probability for $2r$ -round SPN cipher, $p_d^{(2r)}$, is upper bounded by $p_s^{(rB_d)}$.*

From the duality between differential characteristics and linear approximations [4,14], the following definition and theorem also are established.

Definition 6. *The linear branch number B_l is defined as:*

$$B_l = \min_{\Gamma Y \neq 0} (H_w(\theta^*(\Gamma Y)) + H_w(\Gamma Y)),$$

where ΓY is an output mask value of the diffusion layer θ and $\theta^*(\Gamma Y)$ is an input mask value of the layer. θ^* is the diffusion function of mask values concerning the layer.

Theorem 2. *The maximum linear characteristic probability for $2r$ -round SPN cipher, $q_l^{(2r)}$, is upper bounded by $q_s^{(rB_l)}$.*

4 Upper Bound of Differential Characteristic Probability

In this section, we investigate the upper bound of differential characteristic probability of Feistel cipher with SPN round function. In the same way as in the previous section, our goal is to clarify the (lower bound of) minimum number of differential active s -boxes in some consecutive rounds of Feistel cipher.

First, we show the useful lemma concerning the hamming weight for Feistel ciphers.

Lemma 1. *In Feistel ciphers, the following relationship holds.*

$$H_w(\Delta Y^{(i)}) = H_w(\Delta X^{(i-1)} \oplus \Delta X^{(i+1)}) \leq H_w(\Delta X^{(i-1)}) + H_w(\Delta X^{(i+1)})$$

Proof.

$$\begin{aligned} H_w(\Delta Y^{(i)}) &= H_w(\Delta X^{(i-1)} \oplus \Delta X^{(i+1)}) \\ &= \#\{s | \Delta x_s^{(i-1)} \neq 0 \text{ and } \Delta x_s^{(i+1)} = 0\} \\ &\quad + \#\{t | \Delta x_t^{(i-1)} = 0 \text{ and } \Delta x_t^{(i+1)} \neq 0\} \\ &\quad + \#\{u | \Delta x_u^{(i-1)} \neq 0 \text{ and } \Delta x_u^{(i+1)} \neq 0 \text{ and } x_u^{(i-1)} \neq x_u^{(i+1)}\} \\ &\leq H_w(\Delta X^{(i-1)}) + \#\{t | \Delta x_t^{(i-1)} = 0 \text{ and } \Delta x_t^{(i+1)} \neq 0\} \\ &\leq H_w(\Delta X^{(i-1)}) + H_w(\Delta X^{(i+1)}) \end{aligned}$$

Q.E.D.

Since there is a linear transformation layer (P -function) in the SPN round function, we will define the differential branch number \mathcal{P}_d in the same way as in the previous section. Note that it is obvious that if S -function is bijective then $H_w(\Delta X) = H_w(\Delta Z)$, since Δz_i also becomes a non-zero output difference through the differential active s_i -box.

Definition 7. If S -function is bijective, the differential branch number \mathcal{P}_d is defined as follows.

$$\mathcal{P}_d = \min_{\Delta X \neq 0} (H_w(\Delta X) + H_w(\Delta Y))$$

Here, we will define the upper bound of the maximum differential characteristic probability of Feistel cipher with SPN round function in the same way as used for the SPN cipher. That is, the upper bound of the probability is shown by the (lower bound of) minimum number of differential active s -boxes.

Definition 8. Assume Feistel cipher with SPN round function. Let $H_w(\Delta X^{(i)})$ be the number of the i th-round differential active s -boxes, then the differential characteristic probability of the r -round Feistel cipher, $p_d^{(r)}$, satisfies the following relationship.

$$p_d^{(r)} \leq p_s^{\min_{(\Delta X^{(0)}, \Delta X^{(1)}, \dots, \Delta X^{(r+1)}) \neq (0, 0, \dots)} \sum_{i=1}^r H_w(\Delta X^{(i)})}$$

From this definition, clarifying the upper bound of the maximum differential characteristic probability becomes equivalent to showing the (lower bound of) minimum number of differential active s -boxes. To discuss the minimum number easily after this, it is denoted as follows.

$$\mathcal{D}^{(r)} = \min_{(\Delta X^{(0)}, \Delta X^{(1)}, \dots, \Delta X^{(r+1)}) \neq (0, 0, \dots)} \sum_{i=1}^r H_w(\Delta X^{(i)})$$

Hereafter, because of limitations of space, we assume P -function is bijective. Note that this leads to $\mathcal{P}_d \geq 2$.

Lemma 2. The minimum number of differential active s -boxes in any three consecutive rounds satisfies $\mathcal{D}^{(3)} \geq 2$.

Proof. If $\Delta X^{(i)} = 0$, then $\Delta Y^{(i)} = 0$ and $\Delta X^{(i-1)} = \Delta X^{(i+1)} \neq 0$. This leads to $\mathcal{D}_1^{(3)} = 2 \times H_w(\Delta X^{(i-1)}) \geq 2$. On the other hand, If $\Delta X^{(i)} \neq 0$, it follows that $\mathcal{D}_2^{(3)} \geq H_w(\Delta X^{(i)}) + H_w(\Delta Y^{(i)}) \geq \mathcal{P}_d$, since Lemma 1 shows $H_w(\Delta X^{(i-1)}) + H_w(\Delta X^{(i+1)}) \geq H_w(\Delta Y^{(i)})$.

Q.E.D.

Lemma 3. The minimum number of differential active s -boxes in any four consecutive rounds satisfies $\mathcal{D}^{(4)} \geq \mathcal{P}_d$.

Proof. Without loss of generality, we assume that the four consecutive rounds run from the first round to the fourth round.

At no time do both input differences into any consecutive two rounds equal zero. In addition, by the assumption, at no time also do both input differences of every two rounds equal zero. Thus we only consider the six following cases concerning input differences into the consecutive four rounds.

$$(1) \Delta X^{(1)} \neq 0, \Delta X^{(2)} \neq 0, \Delta X^{(3)} \neq 0, \Delta X^{(4)} \neq 0$$

- (2) $\Delta X^{(1)} = 0, \Delta X^{(2)} \neq 0, \Delta X^{(3)} \neq 0, \Delta X^{(4)} \neq 0$
- (3) $\Delta X^{(1)} \neq 0, \Delta X^{(2)} = 0, \Delta X^{(3)} \neq 0, \Delta X^{(4)} \neq 0$
- (4) $\Delta X^{(1)} \neq 0, \Delta X^{(2)} \neq 0, \Delta X^{(3)} = 0, \Delta X^{(4)} \neq 0$
- (5) $\Delta X^{(1)} \neq 0, \Delta X^{(2)} \neq 0, \Delta X^{(3)} \neq 0, \Delta X^{(4)} = 0$
- (6) $\Delta X^{(1)} = 0, \Delta X^{(2)} \neq 0, \Delta X^{(3)} \neq 0, \Delta X^{(4)} = 0$

In case (1), by Lemma 2, $\mathcal{D}_1^{(4)} = \mathcal{D}_2^{(3)} + H_w(\Delta X^{(4)}) \geq \mathcal{P}_d + H_w(\Delta X^{(4)}) \geq \mathcal{P}_d + 1$.

In case (2), $\Delta X^{(1)} = 0$ leads to $\Delta Y^{(2)} = \Delta X^{(3)}$. Thus, $\mathcal{D}_2^{(4)} = H_w(\Delta X^{(2)}) + H_w(\Delta Y^{(2)}) + H_w(\Delta X^{(4)}) \geq \mathcal{P}_d + H_w(\Delta X^{(4)}) \geq \mathcal{P}_d + 1$.

Similarly, in cases (3), (4), and (5), we get $\mathcal{D}^{(4)} \geq \mathcal{P}_d + 1$.

In case (6), by Lemma 2, $\mathcal{D}_6^{(4)} = \mathcal{D}_2^{(3)} \geq \mathcal{P}_d$.

Q.E.D.

From the above proof, the following corollary is obtained.

Corollary 1. *The minimum number of differential active s-boxes in any four consecutive rounds satisfies*

(i) $\mathcal{D}^{(4)} \geq \mathcal{P}_d$, if and only if the input differences in both the first round and the fourth round are zero.

(ii) $\mathcal{D}^{(4)} \geq \mathcal{P}_d + 1$ in the other cases.

Lemma 4. *The minimum number of differential active s-boxes in any six consecutive rounds satisfies $\mathcal{D}^{(6)} \geq \mathcal{P}_d + 2$.*

Proof. – If $\Delta X^{(2)} \neq 0$ and $\Delta X^{(5)} \neq 0$, by Lemma 2, $\mathcal{D}_1^{(6)} = \mathcal{D}_2^{(3)} + \mathcal{D}_2^{(3)} \geq 2 \times \mathcal{P}_d$.

– If $\Delta X^{(2)} = \Delta X^{(5)} = 0$, we get $\Delta X^{(1)} = \Delta X^{(3)}$ and $\Delta Y^{(3)} = \Delta X^{(4)} = \Delta X^{(6)}$. Thus, $\mathcal{D}_2^{(6)} = 2 \times (H_w(\Delta X^{(3)}) + H_w(\Delta X^{(4)})) = 2 \times (H_w(\Delta X^{(3)}) + H_w(\Delta Y^{(3)})) \geq 2 \times \mathcal{P}_d$

– If $\Delta X^{(2)} = 0$ and $\Delta X^{(5)} \neq 0$, or $\Delta X^{(2)} \neq 0$ and $\Delta X^{(5)} = 0$, then $\mathcal{D}_3^{(6)} = \mathcal{D}_1^{(3)} + \mathcal{D}_2^{(3)} \geq \mathcal{P}_d + 2$ by Lemma 2.

Q.E.D.

Lemma 5. *The minimum number of differential active s-boxes in any eight consecutive rounds satisfies $\mathcal{D}^{(8)} \geq 2 \times \mathcal{P}_d + 1$.*

Proof. Again, corollary 1 shows that, in any four consecutive rounds, the minimum number of differential active s-boxes satisfies (i) $\mathcal{D}^{(4)} \geq \mathcal{P}_d$, if and only if the input differences in both the first round and the fourth round are zero, and $\mathcal{D}^{(4)} \geq \mathcal{P}_d + 1$ in the other cases.

Since there is no case in which both input differences into any two consecutive rounds are zero at the same time, the input differences in both the fourth and fifth rounds cannot be zero. That is, the eight consecutive rounds cannot be divided into two cases (i). Thus, $\mathcal{D}^{(8)} \geq \mathcal{P}_d + (\mathcal{P}_d + 1) \geq 2 \times \mathcal{P}_d + 1$.

Q.E.D.

Lemma 6. *The minimum number of differential active s -boxes in any twelve consecutive rounds satisfies $\mathcal{D}^{(12)} \geq 3 \times \mathcal{P}_d + 1$.*

Proof. $\mathcal{D}^{(12)}$ can be converted to three expressions, i.e., $4 \times \mathcal{D}^{(3)}$, $2 \times \mathcal{D}^{(6)}$, and $\mathcal{D}^{(8)} + \mathcal{D}^{(4)}$. Since $\mathcal{D}^{(12)}$ satisfies the three evaluations at the same time, $\mathcal{D}^{(12)} = \max\{4 \times \mathcal{D}^{(3)}, 2 \times \mathcal{D}^{(6)}, \mathcal{D}^{(8)} + \mathcal{D}^{(4)}\} \geq \mathcal{D}^{(8)} + \mathcal{D}^{(4)} \geq 3 \times \mathcal{P}_d + 1$.

Q.E.D.

From the proofs of above-mentioned lemmas, the useful theorem for the $4r$ -round Feistel ciphers is established as follows.

Theorem 3. *The minimum number of differential active s -boxes $\mathcal{D}^{(4r)}$ for $4r$ -round Feistel ciphers with SPN round function satisfies $\mathcal{D}^{(4r)} \geq r \times \mathcal{P}_d + \lfloor r/2 \rfloor$.*

Knudsen argued that for a Feistel cipher to be practically secure against differential and linear cryptanalyses, the upper bounds of the maximum differential and linear characteristic probabilities must be less than the security threshold. Generally speaking, the security threshold is equated to the inverse of the number of all plaintext blocks, i.e., 2^{-64} for 64-bit ciphers and 2^{-128} for 128-bit ciphers.

For example, let the maximum differential probability of an 8-bit s -box be $p_s = 2^{-6}$ and the differential branch number be $\mathcal{P}_d = 5$. It follows that 18-round Feistel ciphers, such as Camellia [1], are practically secure against differential cryptanalysis because of the following corollary.

Corollary 2. *Assuming that the round function consists of s -boxes yielding the maximum differential probability $p_s = 2^{-6}$ and P -function yielding the differential branch number $\mathcal{P}_d = 5$, then a 128-bit Feistel cipher with more than 16-rounds has no effective differential characteristic.*

Proof. By Definition 8 and Theorem 3, $p_d^{(16)} \leq (2^{-6})^{4 \times 5 + 2} = 2^{-132} < 2^{-128}$.

Q.E.D.

5 Upper Bound of Linear Characteristic Probability

In this section, the upper bound of linear characteristic probability is derived in the same way as in the previous section. That is, our goal is to clarify the (lower bound of) minimum number of linear active s -boxes in some consecutive rounds of Feistel cipher using the duality of differential characteristic and linear approximation.

First, the following theorem is established.

Theorem 4. *Consider a Feistel cipher with SPN round function. If the linear transformation layer P (P -function) is bijective, the cipher can be transformed into a Feistel cipher with the PSN round function.*

Proof. From the assumption that P -function is bijective, let describe $P(Z)$ as the transformation of Z by the P -function, and $P^{-1}(Z)$ as that by the inverse function of the P -function.

As mentioned above, in a Feistel cipher with SPN round function, the equation, $X^{(i+1)} = X^{(i-1)} \oplus P(S(X^{(i)}))$, is satisfied. Now, let $V^{(i)} = P^{-1}(X^{(i)})$. The above equation can be transformed as follows, since $C = A \oplus P(B) \Leftrightarrow C = P(P^{-1}(A) \oplus B)$ for any (A, B, C) .

$$\begin{aligned} X^{(i+1)} = X^{(i-1)} \oplus P(S(X^{(i)})) &\Leftrightarrow X^{(i+1)} = P(P^{-1}(X^{(i-1)}) \oplus S(X^{(i)})) \\ &\Leftrightarrow P(V^{(i+1)}) = P(V^{(i-1)} \oplus S(P(V^{(i)}))) \\ &\Leftrightarrow V^{(i+1)} = V^{(i-1)} \oplus S(P(V^{(i)})) \end{aligned}$$

The equation, $V^{(i+1)} = V^{(i-1)} \oplus S(P(V^{(i)}))$, denotes a Feistel cipher with the PSN round function. Accordingly, the ciphertext $(X^{(r)}, X^{(r+1)})$ obtained by applying a Feistel cipher with SPN round function to a plaintext $(X^{(1)}, X^{(0)})$ is equivalent to the result of changing the plaintext $(X^{(1)}, X^{(0)})$ to $(V^{(1)}, V^{(0)})$ by the P^{-1} -function first, then getting $(V^{(r)}, V^{(r+1)})$ from $(V^{(1)}, V^{(0)})$ from the Feistel cipher with PSN round function, and finally transforming it into the ciphertext $(X^{(r)}, X^{(r+1)})$ by the P -function.

Q.E.D.

Starting with the duality between differential characteristic and linear approximation, we will define the linear branch number \mathcal{P}_l , which is similar to the differential branch number \mathcal{P}_d . Hereafter, we assume P -function is bijective.

Definition 9. The linear branch number \mathcal{P}_l is defined as:

$$\mathcal{P}_l = \min_{\Gamma Y \neq 0} (H_w(P^*(\Gamma Y)) + H_w(\Gamma Y)) = \min_{\Gamma Y \neq 0} (H_w(\Gamma Z) + H_w(\Gamma Y)),$$

where ΓY , ΓZ is an output mask value and an input mask value of the P -function, respectively, and P^* is a diffusion function of mask values concerning the P -function.

Next, we will define the upper bound of the linear characteristic probability of a Feistel cipher with SPN round function. That is, the upper bound of the probability is shown by the (lower bound of) minimum number of linear active s -boxes.

Definition 10. Assume a Feistel cipher with SPN round function. If $H_w(\Gamma Z^{(i)})$ is the number of the i th-round linear active s -boxes, then the linear characteristic probability of the r -round Feistel cipher satisfies the following relationship.

$$p_l^{(r)} \leq p_s^{\min_{(\Gamma Y^{(0)}, \dots, \Gamma Y^{(r)}, \Gamma Y^{(r+1)}) \neq (\dots, 0, 0)} \sum_{i=1}^r H_w(\Gamma Z^{(i)})},$$

where $\Gamma Z^{(i)} = P^*(\Gamma Y^{(i)})$ and P^* is the diffusion function of mask values concerning the P -function.

From this definition, clarifying the upper bound of the linear characteristic probability becomes equivalent to determining the (lower bound of) minimum number of linear active s -boxes. To discuss the minimum number easily after this, we denote it as follows.

$$\mathcal{L}^{(r)} = \min_{(\Gamma Y^{(0)}, \dots, \Gamma Y^{(r)}, \Gamma Y^{(r+1)}) \neq (\dots, 0, 0)} \sum_{i=1}^r H_w(\Gamma Z^{(i)})$$

Theorem 5. *Assume a Feistel cipher with SPN round function. If both S -function and P -function are bijective, then $\mathcal{L}^{(r)}$ and \mathcal{P}_l also satisfy Lemma 2 to Lemma 6 and Theorem 3.*

Proof. Because of the bijective P -function, a Feistel cipher with SPN round function is transformed into one with PSN round function by Theorem 4. The cipher can be described as:

$$V^{(i+1)} = V^{(i-1)} \oplus S(P(V^{(i)})) = V^{(i-1)} \oplus S(X^{(i)}) = V^{(i-1)} \oplus Z^{(i)},$$

where $V^{(i)} = P^{-1}(X^{(i)})$, $Z^{(i)} = S(X^{(i)})$.

From the duality between differential characteristic and linear approximation, the linear approximation of the round function of the transformed cipher can be expressed as follows using the concatenation rules [4,14].

$$\Gamma V^{(i)} = \Gamma Z^{(i-1)} \oplus \Gamma Z^{(i+1)} = P^*(\Gamma X^{(i)})$$

By the way, since S -function is bijective, $H_w(\Gamma X) = H_w(\Gamma Z)$ because Γx_i is a non-zero input mask value of a linear active s_i -box. Therefore, the linear branch number \mathcal{P}_l is redefined as:

$$\mathcal{P}_l = \min_{\Gamma X \neq 0} (H_w(P^*(\Gamma X)) + H_w(\Gamma X)) = \min_{\Gamma Z \neq 0} (H_w(\Gamma V) + H_w(\Gamma Z))$$

Accordingly, if $\Delta X^{(i)}$ and $\Delta Y^{(i)}$ are exchanged for $\Gamma Z^{(i)}$ and $\Gamma V^{(i)}$, respectively, it turns out that all proofs are satisfied in the same way as for Lemma 2 to Lemma 6 and Theorem 3.

Q.E.D.

For example, the P^* -function of Camellia can be expressed as:

$$P_{Camellia}^* = \begin{bmatrix} 0 & 1 & 1 & 1 & 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 & 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 & 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 & 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 & 1 & 1 & 1 \end{bmatrix}$$

Thus, it is easily seen that $\mathcal{P}_l = 5$, and the following corollary is obtained.

Corollary 3. *Camellia with reduced to 16-rounds (without FL - and FL^{-1} -functions) has no effective linear approximation.*

Proof. The maximum linear probability of Camellia's s -boxes is $q_s = 2^{-6}$. From Theorem 5 and $\mathcal{P}_l = 5$, the maximum linear characteristic probability of Camellia with reduced to 16-rounds is also upper bounded by 2^{-132} .

Q.E.D.

6 Necessary Condition for Desirable P -Functions

In this section, we consider the necessary condition for desirable P -functions. Here, “desirable” means that the round functions are invulnerable to linear cryptanalysis as well as differential cryptanalysis.

Obviously, the condition is $\mathcal{P}_d = \mathcal{P}_l$ from Sect. 4 and Sect. 5. Thus, we investigate P -functions wherein $\mathcal{P}_d = \mathcal{P}_l$.

Theorem 6. *Assume that P -function is bijective and is expressed as an $n \times n$ matrix P over $\text{GF}(2)^m$. When the P -function satisfies $[y_i]^t = [p_{ij}][z_j]^t$, the following relations are satisfied.*

$$[\Delta y_i]^t = [p_{ij}][\Delta z_j]^t, \quad [\Gamma z_i]^t = [p_{ij}]^t[\Gamma y_j]^t = [p_{ji}][\Gamma y_j]^t,$$

where $[x_i]$ denotes the vector (or matrix) of X and $[x_i]^t$ denotes the transposed vector (or matrix) of X .

Proof. First, since $y_i = \bigoplus_{j=1}^n (p_{ij} \cdot z_j)$,

$$\begin{aligned} \Delta y_i &= y_i \oplus y'_i = \left(\bigoplus_{j=1}^n (p_{ij} \cdot z_j) \right) \oplus \left(\bigoplus_{j=1}^n (p_{ij} \cdot z'_j) \right) \\ &= \bigoplus_{j=1}^n (p_{ij} \cdot z_j \oplus p_{ij} \cdot z'_j) \\ &= \bigoplus_{j=1}^n (p_{ij} \cdot (z_j \oplus z'_j)) = \bigoplus_{j=1}^n (p_{ij} \cdot \Delta z_j) \end{aligned}$$

Thus, $[\Delta y_i]^t = [p_{ij}][\Delta z_j]^t$ is satisfied.

Second, since the P -function is bijective, $Z \cdot \Gamma Z = Y \cdot \Gamma Y$. Then,

$$\begin{aligned} Y \cdot \Gamma Y &= \bigoplus_{j=1}^n \left(\left(\bigoplus_{i=1}^n (p_{ji} \cdot z_i) \right) \cdot \Gamma y_j \right) = \bigoplus_{j=1}^n \left(\bigoplus_{i=1}^n (p_{ji} \cdot z_i \cdot \Gamma y_j) \right) \\ &= \bigoplus_{i=1}^n \left(\bigoplus_{j=1}^n ((p_{ji} \cdot \Gamma y_j) \cdot z_i) \right) = \bigoplus_{i=1}^n \left(\left(\bigoplus_{j=1}^n (p_{ji} \cdot \Gamma y_j) \right) \cdot z_i \right) \end{aligned}$$

On the other hand, since $Z \cdot \Gamma Z = \bigoplus_{i=1}^n (z_i \cdot \Gamma z_i)$, it is obvious that $\Gamma z_i = \bigoplus_{j=1}^n (p_{ji} \cdot \Gamma y_j) = \bigoplus_{j=1}^n (p_{ij}^t \cdot \Gamma y_j)$. Thus, $[\Gamma z_i]^t = [p_{ji}][\Gamma y_j]^t = [p_{ij}]^t[\Gamma y_j]^t$ is satisfied.

Q.E.D.

Theorem 7. Assume that I' is a set of matrices that consist of only one 1-element and $(m-1)$ 0-elements in each line and row, i.e., the matrices generated by only interchanging lines and/or rows of unit matrix.

If a bijective P -function can be expressed as an $n \times n$ matrix P over $\text{GF}(2)^m$ such that $P^t \cdot P \in I'$ or $P^t = I_2 \cdot P \cdot I_1$ where $I_1, I_2 \in I'$, then the P -function satisfies $\mathcal{P}_d = \mathcal{P}_l$.

Proof. By Theorem 6, if the P -function can be expressed as an $n \times n$ matrix P over $\text{GF}(2)^m$, then

$$\mathcal{P}_d = \min_{\Delta Z \neq 0} (H_w(\Delta Z) + H_w(P(\Delta Z))), \quad \mathcal{P}_l = \min_{\Gamma Y \neq 0} (H_w(P^t(\Gamma Y)) + H_w(\Gamma Y))$$

(i) In the case of $P^t \cdot P = I^* \in I'$, let $\Gamma Y = P(\Gamma W)$.

Since the P -function is bijective, it is guaranteed that $\{\Gamma Y\} = \{\Gamma W\}$. Thus,

$$\begin{aligned} \mathcal{P}_l &= \min_{\Gamma W \neq 0} (H_w(P^t(P(\Gamma W))) + H_w(P(\Gamma W))) \\ &= \min_{\Gamma W \neq 0} (H_w(I^* \cdot \Gamma W) + H_w(P(\Gamma W))). \end{aligned}$$

Here, because $I^* \in I'$, $I^* \cdot \Gamma W$ leads to another vector simply by interchanging the elements of ΓW . Thus, $H_w(\Gamma W) = H_w(I^* \cdot \Gamma W)$. As a result, $\exists(\Delta Z, \Gamma W)$, s.t. $\mathcal{P}_d = \mathcal{P}_l$.

(ii) In the case of $P^t = I_2 \cdot P \cdot I_1$ where $I_1, I_2 \in I'$, as mentioned above, since I_1 and I_2 lead to another vector simply by interchanging the elements, $H_w(\Delta X) = H_w(I_1(\Delta X))$ and $H_w(I_2(\Gamma W)) = H_w(\Gamma W)$. Now, let $\Delta Z = I_1(\Delta X)$. Since I_1 is bijective, it is guaranteed that $\{\Delta X\} = \{\Delta Z\}$. Thus,

$$\begin{aligned} \mathcal{P}_d &= \min_{\Delta X \neq 0} (H_w(I_1(\Delta X)) + H_w(P \cdot I_1(\Delta X))) \\ &= \min_{\Delta X \neq 0} (H_w(\Delta X) + H_w(P \cdot I_1(\Delta X))). \end{aligned}$$

On the other hand,

$$\begin{aligned} \mathcal{P}_l &= \min_{\Gamma Y \neq 0} (H_w(I_2 \cdot P \cdot I_1(\Gamma Y)) + H_w(\Gamma Y)) \\ &= \min_{\Gamma Y \neq 0} (H_w(P \cdot I_1(\Gamma Y)) + H_w(\Gamma Y)). \end{aligned}$$

As a result, $\exists(\Delta X, \Gamma Y)$, s.t. $\mathcal{P}_d = \mathcal{P}_l$.

Q.E.D.

For example, the relationship between P -function and P^* -function of Camellia is shown as follows. Thus Theorem 7 indicates that the P -function of Camellia is “desirable.”

$$P_{\text{Camellia}}^* = P_{\text{Camellia}}^t = I^* \cdot P_{\text{Camellia}} \cdot I^*,$$

because

$$P_{Camellia} = \begin{bmatrix} 0 & 1 & 1 & 1 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 1 & 1 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 & 1 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 & 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 & 1 & 1 & 0 & 1 \end{bmatrix}, \quad P_{Camellia}^* = \begin{bmatrix} 0 & 1 & 1 & 1 & 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 & 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 & 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 & 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 & 1 & 1 & 1 \end{bmatrix}, \quad I^* = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \end{bmatrix}$$

7 Conclusion

This paper studied the upper bounds of the maximum differential and linear characteristic probabilities of Feistel ciphers with SPN round function. In the same way as for SPN ciphers, we considered the minimum number of differential and linear active s -boxes, which are a measure of the upper bounds of these probabilities, in order to evaluate security against differential and linear cryptanalyses. The advantage of this method is that it considers the interrelation between input and output data in consecutive rounds, unlike Knudsen's estimation.

We focused on the minimum number of active s -boxes in some consecutive rounds of Feistel ciphers, i.e., in three, four, six, eight, and twelve consecutive rounds, since they can determine the upper bounds of the maximum differential and linear probabilities using the differential and linear branch numbers \mathcal{P}_d , \mathcal{P}_l , respectively. These numbers provide the avalanche effects of P -functions with regard to differential and linear characteristics. As a result, we clarified that the lower bounds of the minimum number of differential (resp. linear) active s -boxes are 2, \mathcal{P}_d (\mathcal{P}_l), $\mathcal{P}_d + 2$ ($\mathcal{P}_l + 2$), $2\mathcal{P}_d + 1$ ($2\mathcal{P}_l + 1$), and $3\mathcal{P}_d + 1$ ($3\mathcal{P}_l + 1$), respectively. The interesting result is that the lower bound of the minimum number of active s -boxes is proportional to the branch number every fourth round, while it seems to be every third round at first glance. Furthermore, this means that, if the branch number is the same, a $2r$ -round Feistel cipher has almost same invulnerability to differential and linear cryptanalyses as a r -round SPN cipher in terms of the upper bounds of the maximum differential and linear probabilities.

Finally, we investigated the necessary condition for desirable P -functions, which means that the round functions are invulnerable to both differential and linear cryptanalyses. In addition, we showed the example of the round function of Camellia, which satisfies the condition.

References

1. K. Aoki, T. Ichikawa, M. Kanda, M. Matsui, S. Moriai, J. Nakajima, and T. Tokita, "Camellia: A 128-bit Block Cipher Suitable for Multiple Platforms – Design and Analysis –," *Selected Areas in Cryptography — 7th Annual International Workshop, SAC2000*, LNCS in this proceeding.

2. K. Aoki, K. Kobayashi, and S. Moriai, "Best Differential Characteristic Search of FEAL," *Fast Software Encryption — 4th International Workshop, FSE'97*, LNCS **1267**, pp.41–53, 1997.
3. K. Aoki, and K. Ohta, "Strict Evaluation of the Maximum Average of Differential Probability and the Maximum Average of Linear Probability," *IEICE Transactions Fundamentals of Electronics, Communications and Computer Sciences*, Vol. E80-A, No. 1, pp. 2–8, 1997.
4. E. Biham, "On Matsui's Linear Cryptanalysis," *Advances in Cryptology — EUROCRYPT'94*, LNCS **950**, pp.341–355, 1995.
5. E. Biham, R. Anderson, and L. R. Knudsen, "Serpent: A New Block Cipher Proposal," *Fast Software Encryption — 5th International Workshop, FSE'98*, LNCS **1372**, pp.222–238, 1998.
6. E. Biham, and A. Shamir, "Differential cryptanalysis of DES-like cryptosystems," *Journal of Cryptology*, Vol.4, No.1, pp.3–72, 1991.
7. Data Encryption Standard, FIPS-PUB-46, 1977.
8. J. Daemen, L. Knudsen, and V. Rijmen, "The block cipher SQUARE," *Fast Software Encryption — 4th International Workshop, FSE'97*, LNCS **1267**, pp.54–68, 1997.
9. S. Hong, S. Lee, J. Lim, J. Sung, and D. Cheon, "Provable Security against Differential and Linear Cryptanalysis for the SPN structure," *Fast Software Encryption Workshop 2000*, 2000. (LNCS to appear).
10. L. R. Knudsen, "Practically Secure Feistel Ciphers," *Fast Software Encryption — Cambridge Security Workshop*, LNCS **809**, pp.211–221, 1994.
11. M. Kanda, Y. Takashima, T. Matsumoto, K. Aoki, and K. Ohta, "A strategy for constructing fast round functions with practical security against differential and linear cryptanalysis," *Selected Areas in Cryptography — 5th Annual International Workshop, SAC'98*, LNCS **1556**, pp.264–279, 1999.
12. X. Lai, J. L. Massey, and S. Murphy, "Markov ciphers and differential cryptanalysis," *Advances in Cryptology — EUROCRYPT'91*, LNCS **547**, pp.17–38, 1991.
13. M. Matsui, "Linear cryptanalysis method for DES cipher," *Advances in Cryptology — EUROCRYPT'93*, LNCS **765**, pp.386–397, 1994.
14. M. Matsui, "On Correlation Between the Order of S-boxes and the Strength of DES," *Advances in Cryptology — EUROCRYPT'94*, LNCS **950**, pp.366–375, 1995.
15. M. Matsui, "New Block Encryption Algorithm MISTY," *Fast Software Encryption — 4th International Workshop, FSE'97*, LNCS **1267**, pp.54–68, 1997.
16. S. Moriai, K. Aoki, and K. Ohta, "The Best Linear Expression Search of FEAL," *IEICE Transactions Fundamentals of Electronics, Communications and Computer Sciences*, Vol. E79-A, No. 1, pp. 2–11, 1996.
17. K. Nyberg, "Linear Approximation of Block Ciphers," *Advances in Cryptology — EUROCRYPT'94*, LNCS **950**, pp.439–444, 1995.
18. K. Nyberg, and L. R. Knudsen, "Provable Security Against a Differential Attack," *Journal of Cryptology*, Vol. 8 No. 1, pp. 27–37, 1995.
19. V. Rijmen, J. Daemen, B. Preneel, A. Bosselaers, and E. D. Win, "The cipher SHARK," *Fast Software Encryption — Third International Workshop*, LNCS **1039**, pp.99–111, 1996.

Author Index

- Adams, Carlisle 158
Aoki, Kazumaro 39

Bergadano, Francesco 144
Blundo, Carlo 130
De Bonis, Annalisa 130

Čanda, Valér 89
Carroll, Christopher 1
Cavagnino, Davide 144
Chan, Agnes 1
Crispo, Bruno 144

Dawson, E. 248

Fluhrer, Scott R. 14

Golić, Jovan Dj. 233, 248
Gong, G. 217
Granboulan, Louis 57
Günther, Christian 106

Horváth, Tamás 89
Hwang, Joon Ho 202
Hühnlein, Detlef 275, 288

Ichikawa, Tetsuya 39
Iwata, Tetsu 303

Jacobson, Michael J., Jr. 275

Kanda, Masayuki 39, 324
Kaneko, Toshinobu 315
Kawamura, Shinichi 72
Kurosawa, Kaoru 303

Lange, Tanja 106
Lee, Pil Joong 202

Magliveras, Spyros 89

Masucci, Barbara 130
Matsui, Mitsuru 39
McGrew, David A. 14
Millan, William L. 248
Moriai, Shiho 39
Muratani, Hirofumi 72

Nakajima, Junko 39
Nguyen, Phong Q. 57
Noilhan, Fabrice 57

Ohkuma, Kenji 72

Park, Nan Kyoung 202
Paulus, Sachar 288
Pliam, John O. 169

Quang, Viet Duong 303

Sano, Fumihiko 72
Seki, Haruki 315
Simpson, Leonie Ruth 248
Stein, Andreas 106
Stinson, Douglas R. 130

Tokita, Toshio 39
van Trung, Tran 89

Vaudenay, Serge 57, 189

Weber, Damian 275
Wu, Huapeng 118

Youssef, A.M. 29, 217

Zhang, Muxiang 1
Zhang, Xian-Mo 262
Zheng, Yuliang 262
Zuccherato, Robert 158